Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería técnica de Telecomunicación:
especialidad Telemática

Proyecto Fin de Carrera

Desarrollo del juego educativo

"Contain the Monsters" para móviles y tablets

Autor: Raúl de Santos Rico
Tutor: Predro J. Muñoz Merino
Octubre, 2014

AGRADECIMIENTOS:

Este proyecto quiero dedicárselo en especial a mis padres por el gran esfuerzo y sacrificio que han hecho a lo largo de sus vidas para que tal día como hoy pueda presentar este proyecto y culminar así mi carrera universitaria.
No tengo duda de que sin su apoyo, su preocupación y su continua dedicación, no habría podido llegar hasta aquí.

Con este proyecto culmino un largo camino de satisfacciones y dificultades, de los que deseo que el día de mañana pueda sentirme orgulloso y recoger los frutos sembrados.

Quisiera agradecer también a la gente que he conocido a lo largo de los cursos de la carrera y que en mayor o menor medida, han participado y me han ayudado a atravesar este largo camino.

En particular y en relación con este Proyecto, no quiero terminar sin hacer mención a Nuria, por el apoyo recibido, por creer en mí y por animarme en los momentos donde más necesitaba un empujón de optimismo para sacarlo adelante.

Finalmente quiero agradecer a la Universidad Carlos III de Madrid y a todos los profesores que me han ayudado a pesar de mis circunstancias particulares con el plan antiguo y el plan Bolonia. Quiero hacer especial mención a mi tutor Pedro J. Muñoz Merino, por darme la oportunidad de hacer este proyecto tal y como yo lo concebí y por los consejos y orientaciones en aquellas cuestiones en las que he tenido más dificultad.

Índice:

Capítulo 1: Introducción *1*

1.1 Motivación **1**

1.2 Objetivos **1**

1.3 Planificación **3**

Capítulo 2: Trabajos relacionados *5*

2.1 Introducción al mundo de los videojuegos **5**

2.2 Historia de los videojuegos **6**

2.3 La industria del videojuego en la actualidad **18**

2.4 Los videojuegos educativos **23**

2.5 Ejemplos de videojuegos educativos existentes **26**

2.6 Software existente para la creación y desarrollo de videojuegos **31**

2.7 Unity 3D **36**

2.7.1 ¿Qué es Unity 3D? 36

2.7.2 Criterios para la decisión de Unity 3D 37

2.7.3 Juegos desarrollados con Unity 3D 39

Capítulo 3: Especificación de requisitos *41*

3.1 Recopilando ideas para un videojuego **41**

3.2 Plantillas de especificación de requisitos: **43**

3.3 Casos de uso: **58**

3.4 Elementos y estrategias introducidas en "Contain the Monsters" **61**

Capítulo 4: Diseño e implementación *65*

4.1 Diseño del proyecto y elementos que conforman la Jerarquía. **65**

4.1.1 Objetos permanentes en la Vista de Jerarquía 65

4.1.1.1 Pantalla principal 66

4.1.1.2 Pantalla de juego 71

4.1.1.3 Objetos comunes 80

4.1.2 Objetos prefabs presentes en el proyecto 85

4.1.2.1 Enemigos 85

4.1.2.2 Torratas 96

4.1.2.3 Efectos de partículas 100

4.2 Implementación del proyecto **103**

4.2.1 Lenguaje de programación: 103

4.2.2 Estructura básica de un script en Unity 105

4.2.3 Partes de código más destacables 107

4.2.3.1 IA de los enemigos: funcionamiento de los Nodos. 107

4.2.3.2 IA de los enemigos: Raycast de los enemigos 109
4.2.3.3 Lectura de ficheros XML en Unity 111
4.3 Problemas y soluciones aplicadas **111**
4.3.1 Memoria RAM llena: 111
4.3.2 Bajo frame rate: perspectiva de la cámara 112
4.3.3 NavMesh y sus problemas 114
Capítulo 5: Evaluación y resultados ***116***
5.1 Evaluación **116**
5.2 Resultados **119**
Capítulo 6: Conclusiones y trabajos futuros ***123***
6.1 Conclusiones: **123**
6.2 Trabajos futuros: **124**
Capítulo 7: Presupuesto ***125***
7.1 Recursos materiales **125**
7.2 Recursos humanos **126**
Bibliografía ***127***
ANEXO 1: MANUAL DE USUARIO ***128***
ANEXO 2: MANUAL EDICIÓN DE PREGUNTAS ***129***
ANEXO 3: MANUAL DE INSTALACIÓN ***132***

RESUMEN:

Desde que somos pequeños, se nos enseña una de las formas más atractivas de aprender sin que nos demos cuenta. Esta es a través de los juegos.

Los juegos educativos siempre han estado ahí, desde el principio de los tiempos, en donde se busca una actividad para que los participantes se diviertan y disfruten a la vez que aprenden. Con el avance y la difusión de las nuevas tecnologías, el videojuego se expande como una nueva forma de estimular auditiva y visualmente al jugador, encerrándole en un mundo de diversión y entretenimiento, haciendo de "el logro" como uno de sus objetivos claros, buscando los medios y las formas para llegar a este fin, siendo el reto uno de los elementos dinamizadores de su comportamiento.

De esta forma nace "Contain the Monsters", un juego que presenta un objetivo muy concreto y del que el jugador tendrá que servirse de sus conocimientos y su pericia para poder alcanzar su final.

El jugador se sumerge bajo la piel de un niño que se encuentra solo en una aldea.
Monstruos de diferente índole irán atravesando un camino laberintico desde el principio hasta el final del mapa, hasta llegar a su ansiado objetivo: destruir la aldea.
Para evitar que lleguen, deberemos protegerla usando 2 herramientas fundamentales: verjas de hierro que impedirán que los monstruos vayan avanzando por el laberinto durante un tiempo y torretas automáticas, armas que colocaremos libremente en diferentes puntos del mapa y que permitirán defendernos de las holeadas. Para poder hacernos con estas herramientas, tendremos que valernos de nuestros conocimientos, respondiendo correctamente a las preguntas que se nos muestran a lo largo del camino. El objetivo principal del juego, es que el jugador pueda verse envuelto en la necesidad de contener a los monstruos para que no lleguen, mientras en cada momento te ves en la obligación de poner a prueba tus conocimientos y habilidades.

Para la realización de esta aplicación, se ha empleado la herramienta de creación de juegos Unity3D que permite que, de forma gratuita y sin necesidad de tener grandes conocimientos, podamos crear un videojuego, así como la utilización de JavaScript como lenguaje de programación empleado en los Scripts.

Finalmente, se ha realizado la evaluación de la aplicación resultante a partir de dos encuestas. Un total de 6 preguntas de la primera encuesta servirán para confirmar que la persona encuestada tiene el perfil deseado para responder con objetividad mientras que en la segunda encuesta se evaluará a través de 12 preguntas cómo ha sido la experiencia resultante una vez que se ha probado. Se han obtenido un conjunto de conclusiones positivas de este juego basadas en esta evaluación por encuestas así como a través de las respuestas a preguntas para valorar puntos positivos y negativos del mismo.

ÍNDICE DE FIGURAS

Figura 1: Pantalla de Pong
Figura 2: Anuncio del juego Destruction Dervy para máquinas recreativas
Figura 3: Pantalla del juego Death Race (1976)
Figura 4: Pantalla del famoso Pac Man o Comecocos
Figura 5: Pantalla de Donkey Kong
Figura 6: Pantalla de Tetris
Figura 7: Pantalla inicial de Sonic the Hedgehog
Figura 8: Pantalla de Doom 1
Figura 9: Pantalla de Final Fántasy VII para PSX
Figura 10: Pantalla de Halo
Figura 11: Símbolos utilizados para clasificación de videojuegos por el método PEGI
Figura 12: Pantalla de Perfect Dark Zero
Figura 13: Pantalla de Uncharted 3: Drake's Deception
Figura 14: Snake para móviles
Figura 15: Pantalla de Angy Birds
Figura 16: Pantalla de The Walking Dead para Android
Figura 17: esquema de la cadena de valor tradicional de videojuegos
Figura 18: esquema de la nueva cadena de valor videojuegos
Figura 19: Principales distribuidoras mundiales de videojuegos
Figura 20: Evolución del mercado de videojuegos en España en 2012-2013 (M€)
Figura 21: Evolución del mercado de videojuegos en España en 2012-2013 (miles de unidades)
Figura 22: Porcentaje de respuestas de una muestra de profesores de enseñanza que han utilizado los videojuegos en sus aulas
Figura 23: Pantalla de "Aprende matemáticas con Pipo"
Figura 24: Pantalla de "Darfus is dying"
Figura 25: Una carátula del juego Dragon Box
Figura 26: Pantalla de Hakitzu
Figura 27: Entorno de trabajo de Stencyl
Figura 28: Entorno de trabajo de Game Develop
Figura 29: Entorno de trabajo de Entidad 3D
Figura 30: Entorno de trabajo de Game Develop
Figura 31: Pantalla principal de Game Develop
Figura 32: Ventana de "build settings" de Unity 3D
Figura 33: Diagrama de Casos de Uso
Figura 34: Ejemplo del modo Supervivencia del exitoso juego: Gear of War Judgment
Figura 35: Captura de una versión en desarrollo del proyecto
Figura 36: Modelo 3D de las torretas usadas en el proyecto
Figura 37: Combinación de torretas con vallas en el juego
Figura 38: Cubos de colores que contienen las preguntas
Figura 39: Captura de la ventana de jerarquía de Unity
Figura 40: Captura de la pantalla principal
Figura 41: Jerarquía de "Pantalla Principal"
Figura 42: Jerarquía de "Menu principal"
Figura 43: Captura del menú de la pantalla principal

Figura 44: Captura de la pantalla de juego
Figura 45: Jerarquía de “Pantalla de juego”
Figura 46: Jerarquía de “Cámara gameover”
Figura 47: Jerarquía de “Cámara general”
Figura 48: Implementación del paquete Standard Assets (Mobile)
Figura 49: Jerarquía de “Camera Relative Controls”
Figura 50: Jerarquía del “Player”
Figura 51: Jerarquía de las “GUIs”
Figura 52: Opciones que se despliegan tras pulsar el icono “Menú”
Figura 53: Interfaz de preguntas
Figura 54: Ventana de resultado
Figura 55: Jerarquía de “Objetos comunes”
Figura 56: Jerarquía de “VallasLaberinto”
Figura 57: Línea de colocación de vallas
Figura 58: Objeto “vallaTrigger” renderizado
Figura 59: Jerarquía de “Vida”
Figura 60: Modelo 3D Araña 1
Figura 61: Modelo 3D Araña 2
Figura 62: Modelo 3D Monstruo Verde
Figura 63: Modelo 3D Duende maligno
Figura 64: Modelo 3D Monstruo Azul
Figura 65: Ejemplo de la malla creada por Unity para detectar obstáculos
Figura 66: Mapa del juego con los nodos “pivote” enumerados
Figura 67: Mapa del juego dividido en filas y columnas
Figura 68: Diagramas de estados: estado “Patrulla”
Figura 69: Diagramas de estados: estado “Perseguir”
Figura 70: Diagramas de estados: Estado “Ataque”
Figura 71: Diagramas de estados: Estado Romper
Figura 72: Modelo 3D torreta
Figura 73: Elementos que componen el objeto torreta
Figura 74: Diagramas de flujo de los estados “Vigila” y “Ataca” de la torreta
Figura 75: Representación del sistema de partículas “destello”
Figura 76: Representación del sistema de partículas “impacto”
Figura 77: Representación del sistema de partículas “impacto final”
Figura 78: Representación del sistema de partículas “explosión”
Figura 79: Representación del sistema de partículas “polvo”
Figura 80: Diagrama que explica la estructura de un script en Unity 3D
Figura 81: Representación de los tipos de proyecciones en 2D
Figura 82: Imagen del juego en desarrollo con cámara de proyección en perspectiva
Figura 83: Tutorial de uso del juego
Figura 84: Localización del fichero “plantilla.xml”
Figura 85: Ventana de Build Settings
Figura 86: Contenido de la memoria interna del móvil
Figura 87: Archivo APK de nuestro proyecto
Figura 88: Procediendo a instalar el proyecto en el dispositivo móvil
Figura 89: Instalación del proyecto en el dispositivo móvil
Figura 90: Menú de las aplicaciones instaladas en nuestro móvil

ÍNDICE DE TABLAS:

Tabla 1. Requisito de usuario RU-F001
Tabla 2. Requisito de usuario RU-F002
Tabla 3. Requisito de usuario RU-F003
Tabla 4. Requisito de usuario RU-F004
Tabla 5. Requisito de usuario RU-F005
Tabla 6. Requisito de usuario RU-F006
Tabla 7. Requisito de usuario RU-F007
Tabla 8. Requisito de usuario RU-F008
Tabla 9. Requisito de usuario RU-F009
Tabla 10. Requisito de usuario RU-F010
Tabla 11. Requisito de usuario RU-F011
Tabla 12. Requisito de usuario RU-F012
Tabla 13. Requisito de usuario RU-F013
Tabla 14. Requisito de usuario RU-F014
Tabla 15. Requisito de usuario RU-F015
Tabla 16. Requisito de usuario RU-F016
Tabla 17. Requisito de usuario RU-F017
Tabla 18. Requisito de usuario RU-F018
Tabla 19. Requisito de usuario RU-F019
Tabla 20. Requisito de usuario RU-F020
Tabla 21. Requisito de usuario RU-F021
Tabla 22. Requisito de usuario RU-F022
Tabla 23. Requisito de usuario RU-NF01
Tabla 24. Requisito de usuario RU-NF02
Tabla 25. Requisito de usuario RU-NF03
Tabla 26. Requisito de usuario RU-NF04
Tabla 27. Requisito de usuario RU-NF05
Tabla 28. Requisito de usuario RU-NF06
Tabla 29: Primera tabla modelo de encuesta
Tabla 30: Segunda tabla modelo de encuesta
Tabla 31: Primera tabla de resultados de la encuesta
Tabla 32: Segunda tabla de resultados de la encuesta
Tabla 33: Costes de hardware
Tabla 34: Costes de software
Tabla 35: Recursos materiales
Tabla 36: Recursos materiales
Tabla 37: Presupuesto del Proyecto

Capítulo 1: Introducción

1.1 Motivación

La principal motivación que ha llevado a crear un juego educativo es que los estudiantes puedan evaluar sus conocimientos, aprendiendo a la vez que se divierten y pasan un rato entretenido.

La idea es motivar a los alumnos para que aprendan, jueguen y entren en la dinámica de intentar superarse a sí mismos, utilizando para ello sus conocimientos y sus habilidades. Además, se pretende crear un juego llamativo visualmente, que sea fácil de usar, con objetivos claros y reglas bien definidas y cuyo punto principal sea intentar alcanzar niveles antes no superados. El usuario se podrá familiarizar fácilmente con la jugabilidad, los enemigos que se muestran, así como la forma de impedir que alcancen su objetivo. Podrá evaluar sus conocimientos y aprender, a través de la reiteración, la respuesta correcta a preguntas que antes no conocía para, de esta forma, poder hacerse con los utensilios que le permitan pasar de nivel. Se trata en definitiva de que el jugador decida echar una partida en sus ratos libres y además poner a prueba sus conocimientos, que les serán necesarios para poder superar el juego.

También es de destacar que no solo se ha querido hacer un juego de cara al alumno, sino también al propio profesorado que decida cuál serán las preguntas a formular y de qué materia. Esto permite que el juego sea modificado fácilmente sin necesidad de conocimientos previos por parte de quien quiera adaptar las preguntas que van a ser formuladas.

A las anteriores motivaciones se añade la de que un grupo de jugadores habituales pueda aprobar y evaluar la aplicación positivamente para confirmar la satisfacción del usuario.

1.2 Objetivos

El objetivo principal es idear un juego educativo, diseñarlo y programar toda la lógica y comportamientos que en este intervienen para finalmente implementarlo para un dispositivo táctil. Este juego debe tener las siguientes características:

- Debe ser una aplicación móvil, pensada para un sistema táctil como puede ser un Smartphone o Tablet con sistema operativo Android.

- Su objetivo principal ha de ser divertir y entretener a la vez que usamos el conocimiento como una herramienta fundamental para que el jugador pueda superar los diferentes niveles. Conocimiento y destreza se juntan y serán necesarias para alcanzar nuestro objetivo final.

- Además de ser un juego educativo, tiene que ser un juego que siga los principios básicos de los juegos para máquinas Arcade: diseño sencillo, niveles cortos, controles fáciles de entender y dificultad ascendente con una interrupción mínima entre niveles.

- Debe ser un juego que enganche, que produzca adición y que motive al jugador a seguir jugando.

- Debe ser un juego vistoso, atractivo visualmente y con una buena interfaz.

- Debe ser un juego dinámico, que tenga elementos ya vistos en otros juegos de éxito como es la presencia de monstruos que debemos eliminar.

- Debe poder ser jugado por personas de cualquier edad, niños o mayores, cuya única limitación sea puesta por el docente o persona encargada de editar el fichero que contendrá las preguntas.

- Se debe poder guardar automáticamente la puntuación más alta que alcancemos en el juego para motivarnos e incentivarnos a seguir jugando. De esta forma, cuando volvamos a arrancarlo, podremos verla en la pantalla principal.

- Las preguntas deben poder ser editadas y modificadas por cualquier persona que así lo desee sin necesidad de tenga que tener conocimientos especializados en programación o que tenga que entrar en el código fuente del juego. Bastará por tanto, con editar un fichero XML

- Se evaluará la aplicación resultante y el grado de satisfacción de las personas a partir de una encuesta de 12 preguntas que revelará si se ha conseguido cumplir los objetivos propuestos y cuales son aquellos aspectos a mejorar de la misma.

1.3 Planificación del proyecto y diseño de las diferentes partes que compondrán el juego

Este proyecto tiene como finalidad el crear un videojuego educativo, estilo ARCADE. Esto es, un juego donde la dificultad aumenta progresivamente con cada pantalla superada, pero donde no hay profundidad en cuanto a historia, mitología, personalidad de los personajes, etc... y se utiliza la parte educativa, como una forma de ponernos a prueba ante diversas cuestiones que tendremos que resolver correctamente para ayudarnos a obtener la victoria.

Para realizar esta tarea, haremos uso de Unity 3D, herramienta de programación integrada para la creación de videojuegos 3D y JavaScript como lenguaje de programación, así como XML para leer las preguntas que se van a realizar. Por último, se hará uso también de la herramienta Paint.NET para la edición de iconos o botones que serán usados en nuestro juego.

Las tareas que se han realizado en este trabajo son:

- Diseño de los diferentes elementos que compondrán el juego: escenario, personaje, enemigos, herramientas…

- Crear una interfaz amigable para el usuario, sencilla y que permita ser utilizada para un público con edades diferentes.

- Para la utilización de los diferentes objetos y figuras 3D que veremos en pantalla, hemos hecho uso del **Asset Store** para obtener modelos gratuitos. Esta decisión fue tomada debido a que diseñar modelos propios habría requerido mucho tiempo así como el aprendizaje de modelado en tres dimensiones.

- La implementación se decidió realizar sobre el soporte JavaScript. Unity soporta 3 tipos de lenguaje: JavaScript, C# y Boo, siendo Boo un derivado de Python. Decidimos elegir JavaScript, ya que se trataba de un lenguaje muy similar al que se ha ido estudiando durante la carrera. Además, es el lenguaje más utilizado por la vasta cantidad de tutoriales que hay en internet.

- Diseño del mapa, esto es, del laberinto. Necesitamos diseñar un laberinto que fuese lo suficientemente grande como para darle libertad al jugador y permitir que los monstruos se dispersasen por el mismo.

- Aplicar animaciones a los personajes. Esto es que, para cada movimiento o acción que programemos en los personajes (desplazarse, defenderse, atacar… etc.) se les aplica animaciones que ya vienen incluidas junto con nuestro modelo 3D.

- Programar los diferentes comportamientos de los elementos que intervendrán en el juego así como su inteligencia artificial (IA): personaje principal, enemigos, vayas, torretas y cubos.

- Programar los diferentes estados por los que va pasando el juego. Esto tiene lugar a través del "Controlador del juego" que se encargará de ser quien dicte que ocurrirá en el juego.

- Diseñar los iconos o botones que mostrarán información sobre la partida o bien servirán para que el usuario actúe sobre el juego. Algunos de los dibujos han sido obtenidos de páginas de internet y han sido modificados para adaptarse a la interfaz del juego, recortando partes, redimensionándolos, añadiendo colores…etc. Esto ha sido realizado con el programa gratuito de edición de imagen Paint.NET.

- Programar los diferentes inputs o botones que utilizará el jugador: en la zona derecha el joystick para mover al personaje, en la zona izquierda los botones rojo y amarillo utilizados para posicionar una torreta o valla en el lugar que se encuentra el personaje y arriba a la derecha, el botón de la casa para obtener un vista general del mapa.

- Programar también las diferentes partes que mostrarán información sobre el transcurso de la partida: información sobre el nivel en el que nos encontramos, puntuación que llevamos conseguida, enemigos restantes, estado de salud de nuestra aldea a través de una barra de salud roja, estado de salud de nuestro personaje principal, numero de vallas y torretas acumuladas del que disponemos para colocar.

- Diseñar una pantalla de "GameOver" donde nos indique la puntuación final alcanzada.

- Diseñar un menú principal con el nombre del juego, así como la puntuación máxima alcanzada.

- Implementar la parte educativa y que ésta sea fácilmente modificable por el docente. Basada en lanzar preguntas sobre una determinada materia, el jugador deberá escoger la respuesta correcta para obtener una torreta o una valla. Las preguntas serán editadas y guardadas en un fichero XML almacenado en un directorio del juego, el cual leerá y lanzará en pantalla cada vez que nuestro personaje principal recoja un cubo de color.

- Evaluar la aplicación resultante y el grado de satisfacción de las personas después de probar el juego final a partir de una encuesta de 12 preguntas que revelará si se ha conseguido cumplir los objetivos propuestos y cuales son aquellos aspectos a mejorar en un futuro.

Capítulo 2: Trabajos relacionados

2.1 Introducción al mundo de los videojuegos

El mundo de los videojuegos puede contemplarse desde muy diversos ángulos de los cuales pueden ser el sociológico, el tecnológico, el industrial o el económico como los más importantes.

Según las profesoras Simone Belli y Sara Olivé de la Universidad Autónoma de Barcelona desde el punto de vista sociológico el impacto que los videojuegos han tenido en la sociedad en los últimos treinta años es un fenómeno que, según los propios sociólogos, todavía no se estudiado en profundidad por los investigadores sociales. [1] Desde este punto de vista choca la idea de fomentar el aislamiento con la contraria, que es la ser un elemento más de relación entre personas sobre todo jóvenes. Dada la diversidad en las modalidades de jugar, queda en última instancia la libertad del jugador para tomar una u otra.

Los videojuegos tienen la característica de poder ser jugados en prácticamente cualquier lugar desde la propia casa, lugares públicos e incluso en la calle o en transporte público (si pensamos en videojuegos de móviles). Algunos de estos espacios pueden ser compartidos por varios jugadores, entendiendo estos espacios como lugares para la socialización del conocimiento y lugar donde tener la oportunidad de expresar sus emociones en un marco virtual. La dicotomía es debida a que todavía no se ha encontrado una herramienta adecuada para definir y comprender totalmente esta nueva forma de entretenimiento. Esto podría deberse a que el videojuego es un producto demasiado moderno que puede verse desde un punto de vista individual y colectivo con resultados muy diferentes.

Desde el punto de vista tecnológico, los videojuegos se han desarrollado de la mano de la tecnología aplicada a la informática pero no exclusivamente puesto que el desarrollo de un videojuego es una actividad multidisciplinaria en la que participan profesionales de disciplinas tan variadas como la informática, el diseño gráfico, los efectos de sonido, la música, etc. No vamos a extendernos en este tema que se verá debidamente desarrollado en un apartado posterior pero se puede deducir que la tecnología de los videojuegos posee una gran parte de los estudios científicos de la actualidad.

Si abordamos el tema desde el punto de vista industrial y económico, puesto que ambas facetas están muy relacionadas, la industria de los videojuegos como el sector económico involucrado en el desarrollo, la distribución, la venta de videojuegos y del hardware asociado engloba a docenas de disciplinas de trabajo y emplea a miles de personas alrededor del mundo.

Aunque también se desarrollará en un apartado posterior podemos resumir que la industria de videojuegos ha experimentado en los últimos años un gran crecimiento, debido al desarrollo de la informática, capacidad de procesamiento, imágenes más

reales y la relación entre películas de cine y los videojuegos. En la década de 2000 los videojuegos han pasado a generar más dinero que la del cine y la música juntas, de hecho la industria de videojuegos generó 57.600 millones de euros durante 2009 en todo el mundo. [1]

2.2 Historia de los videojuegos

Sería sumamente difícil dar la fecha en que se inicia nuestra historia. De hecho va unida prácticamente a la misma creación de los computadores puesto que es fácil suponer que los mismos programadores e ingenieros que diseñaron estas máquinas, estuvieran tentados ya casi desde los inicios en hacer trabajar la máquina para su propio entretenimiento y diversión en momentos que sirvieran de relax o de ocio y si ningún fin comercial. No obstante se tuvo enseguida la percepción de que esta forma de entretenimiento podía ser explotada y originar una importante fuente de ingresos.

En 1947 los americanos Thomas T. Goldsmith y Estle Ray Mann simularon una pantalla de radar que detectaba lanzamiento de cohetes aunque por no haber movimiento en la pantalla solo se considera un paso para la creación del verdadero videojuego.

En 1952 el americano Alexander Sandy Douglas crea el famoso “tres en raya” en versión digital. Utilizaba la computadora EDSAC que fue la primer computadora que podía almacenar programas electrónicos. Tampoco tenía movimientos por lo que más bien hay que considerarlo como un juego gráfico.

En 1958 el americano William Higinbotham usa un osciloscopio de laboratorio para crear un juego de tenis llamado “Tennis for Two” que más tarde fue comercializado por Atari con el nombre de “Pong” y que fue un gran éxito.

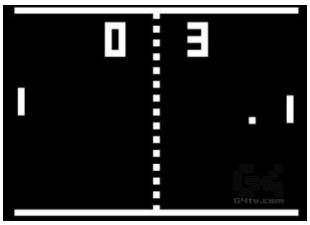


Figura 1: ***Pantalla de Pong***

En la década de los 70 se marca el comienzo auténtico de esta industria incluyendo poco a poco los avances que permiten los nuevos microprocesadores y los chips de memoria incluyéndolos en las videoconsolas y otras máquinas.

De esta década (1975) data la creación de Microsoft por los americanos Bill Gates y Paul Allen y que pronto entró en el negocio de las videoconsolas que se empezaban a extender en lugares públicos como bares, salones recreativos etc.

[1] *Imagen obtenida desde http://www.taringa.net/posts/info/7731318/El-Primer-Videojuego-de-la-Historia.html en Junio de 2014*

Aún no existían los juegos en los domicilios. En este contexto el juego "Destruction Derby" es el primero que incluye violencia en su temática lo que origina las primeras controversias al respecto.

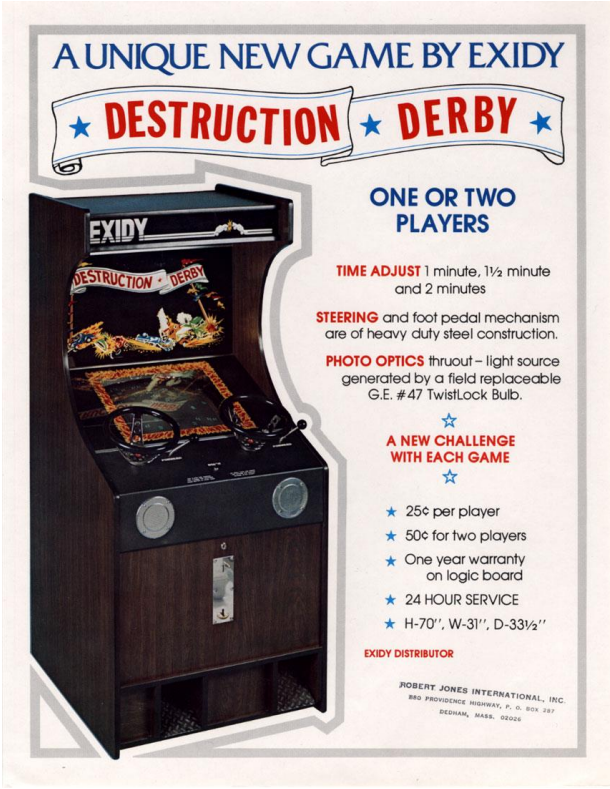


Figura 2: ***Anuncio del juego Destruction Dervy para máquinas recreativas*** [2]

Otros juegos de éxito fueron Death Race (1976), "Space Invaders" y sobre todo el famoso "Asteroids" de la compañía Atari.

En la década de los 80 alcanzan una gran popularidad las máquinas recreativas y las videoconsolas tales como Odissey 2 y Atari 5200.

También aparecen en esta década los primeros ordenadores para el hogar tales como el Commodore 64 y el ZXSpectrum.

[2] *Imagen obtenida desde http://retroinvaders.com/index.php/es/all/5180 en Junio de 2014*

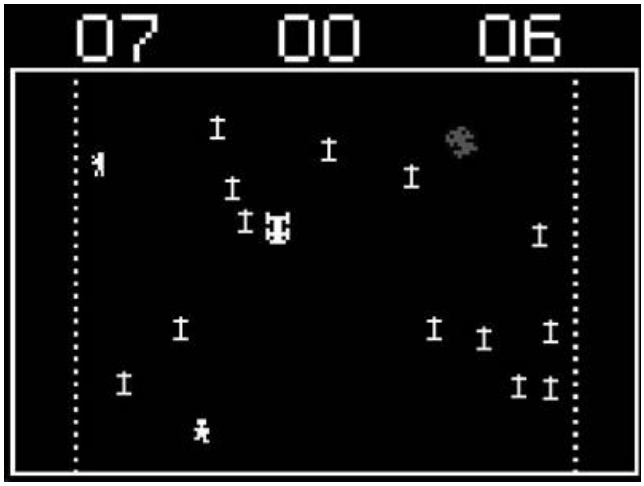


Figura 3: Pantalla del juego Death Race (1976) [3]

Mención aparte merece el juego "Pac Man" de la empresa japonesa Namco. Apareció a principios de la década de los 80 para máquinas recreativas y fue un gran éxito. En España se conoció como "Comecocos" y era simplemente un círculo sin un trozo que al faltar simulaba una boca y el juego consistía en recorrer los laberintos en los que comía puntos de varios tamaños y premios con frutas. Tenía enemigos en forma de fantasmas de diversos tipos que se lo podían comer y terminar el juego. De lo contrario se pasaba al siguiente nivel de los 255 que tenía. Este juego se convirtió en el icono de esa década. La aparición de "Pac Man" abrió el consumo al sexo femenino atraído por el entretenimiento sin violencia. Otra característica era que no se podía ganar a la máquina sino solo conseguir mejor puntuación.

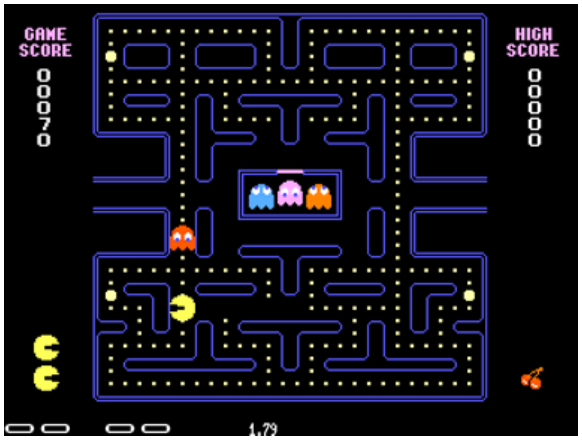


Figura 4: Pantalla del famoso Pac Man o Comecocos [4]

Juegos famosos de la década de los 80 fueron "Donkeykong"(1981), "Pole Position"(1982) y "Zaxxon"(1982) para máquinas recreativas. Estos juegos fueron después adaptados a ordenadores y videoconsolas para el hogar como los mencionados ZXSpectrum y Commodore 64.

---

[3] *Imagen obtenida desde http://www.elmercurio.com.mx/videojuegosagresivosefectoshistoriamuertes-5984.html en Junio de 2014.*

[4] *Imagen obtenida desde http://laestanteriadecho.blogspot.com.es/2012/08/pac-man.html en Junio de 2014*

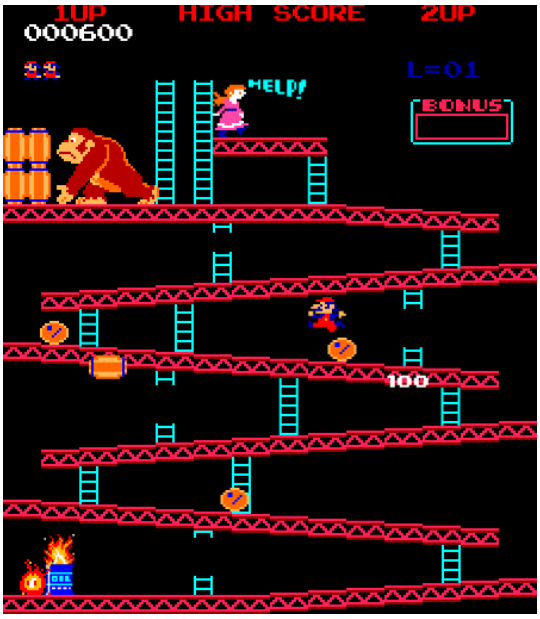


Figura 5: Pantalla de Donkey Kong [5]

En esta década aparecen también nuevas videoconsolas como la N.E.S (Nintendo Entertainment System) en 1983 y que fue un gran éxito en juegos como "Popeye" (1983).

El popular personaje Mario aparece en ese mismo año aunque el que le dio la verdadera popularidad fue "Super Mario Bros" en 1985 de Nintendo. Era el primer juego que tenía un objetivo determinado con un final concreto.

Otro gran triunfo fue el juego "Tetris"(1984) que inventó ruso Alekséi Pázhitnov , cuando trabajaba en la Academia de Ciencias de Moscú y que produjo una fuerte lucha competitiva entre las empresas Atari y Nintendo y fue Nintendo la que al final lo consiguió. Este juego se comercializaría en Europa en 1987 siendo uno de los más vendidos en esta década junto con el mencionado "Pac Man". Era esencialmente un juego de habilidad y reflejos en el que piezas geométricas de distintos tamaños caían a un lugar en la parte inferior de la pantalla donde debían ser encajadas sucesivamente dejando el menor número de huecos posibles entre ellas.

[5] *Imagen obtenida desde http://www.wired.com/2013/03/donkey-kong-pauline-hack/ en Junio de 2014*

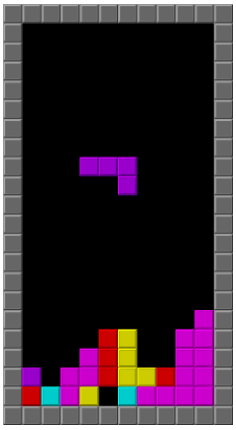


Figura 6: Pantalla de Tetris [6]

Nuevas consolas de 8 y 16 bits aparecen a finales de los años 80 como la Mega Drive de la compañía Sega y aparecen los ordenadores personales con microprocesadores como el Intel 8088 con su propio monitor monocromático y sistema operativo MS-DOS.
En 1989 aparece la consola portátil Game Boy incluyendo el ya mencionado famoso juego "Tetris".

Ya a principios de la década de los 90 se empezaron a comercializar consolas de 16 bits como la Mega Drive de Sega con juegos como "Dragón Ball" o "Fifa 98" o la S.N.E.S. con juegos como "La leyenda de Zelda" o "Super Mario Kart".

Se instaura el soporte CD-ROM que sustituye a los anteriores almacenamientos en cartuchos lo que permite una extraordinaria evolución en consolas y ordenadores.

En 1991 aparece otro personaje que alcanzaría gran renombre, el erizo "Sonic" con gran velocidad y complejidad de movimientos conseguidos gracias al diseño de procesador 68000 de Motorola con una gran velocidad de proceso en comparación con los anteriores. Fue un enorme éxito con la venta de millones de copias.

Figura 7: Pantalla inicial de Sonic the Hedgehog [7]

[6] *Imagen obtenida desde http://es.wikipedia.org/wiki/Tetris en Junio de 2014*
[7] *Imagen obtenida desde http://www.laps3.com/foro/93_trofeos/320962-trofeos_sonic_hedgehog.html en Junio de 2014*

En 1991 también aparece el primer juego en 3 dimensiones "Wolfenstein 3D" con un gran éxito aunque el verdadero primer juego con este entorno fue "Doom" (1993) pudiendo desplazar el personaje con una gran versatilidad. Este juego además abrió la después tan utilizada forma de multijugador online a través de la red o de un módem y que permite además añadir niveles por los propios jugadores.

Figura 8: Pantalla de Doom 1 [8]

Algo más tarde aparece la videoconsola de 32 bits Play Station de Sony y la Sega Saturn así como la Nintendo 64 de 64 bits.

En los ordenadores aparecen las primeras tarjetas aceleradoras que aumentan sensiblemente la calidad y la velocidad de imagen en los juegos con lo que paulatinamente va disminuyendo la afluencia a las máquinas recreativas aunque los fabricantes reaccionarán con máquinas de grandes dimensiones (casi a tamaño real).

La videoconsola para el hogar de más popularidad fue la PlayStation 1 con juegos tan exitosos como "Final Fantasy VII", "Resident Evil" o "Metal Gear Solid".

Con esto entramos en el siglo XXI.

Nuevas consolas aparecen en el año 2001 como la Xbox de Microsoft.

[8] *Imagen obtenida desde http://www.sinjinsolves.com/reviews/360/xbladoom/doom.html en Junio de 2014 en Junio de 2014*

Figura 9: Pantalla de Final Fántasy VII para PSX [9]

Sony lanza la PlayStation 2 que logra ocupar el primer puesto en ventas.

Por otra parte Nintendo lanza la consola Game Boy Advance con juegos como "Final Fantasy" que formará toda una serie de gran éxito.

El mejor juego de ese año fue "Halo" que algunas revistas lo consideraron como uno de los mejores juegos creados nunca.

Figura 10: Pantalla de Halo [10]

La ISFE (Federación de Software Interactivo de Europa) crea en 2003 un método para clasificar los videojuegos según edades e informar si el juego tiene violencia, sexo, racismo, lenguaje grosero, etc… y que fue adoptado por España.

[9] *Imagen obtenida desde http://www.vandal.net/reportaje/especial-final-fantasy/8 en Junio de 2014*
[10] *Imagen obtenida desde http://es.wikipedia.org/wiki/Halo:_Combat_Evolved en Junio de 2014*

Figura 11: Símbolos utilizados para clasificación de videojuegos por el método PEGI [11]

Nintendo lanza en 2003 la consola Game Boy Advance SP con baterías de litio y en 2004 lanza la primera videoconsola con pantalla táctil, la Nintendo DS con micrófono incorporado y conexión wifi.

Sony comercializa la consola PSP tratando de abarcar no solo el mercado de las consolas caseras de sobremesa sino las portátiles. Algunos juegos para esta consola fueron: "FIFA 2006" y "Pro Evolution Soccer 5".

En el año 2005 aparecen las consolas de 7ª generación con la Xbox360 de Mictrosoft incluyendo un lector de películas HD-DVD (Disco Versátil Digital de Alta Densidad).

Algunos juegos de gran calidad para esta consola son: "Perfect Dark Zero", "Call of Duty 3"

Figura 12: Pantalla de Perfect Dark Zero [12]

El primero de ellos permite hasta 32 jugadores simultáneos en red.

[11] *Imagen obtenida desde http://blogs.lainformacion.com/legal-edigital/2012/04/10/son-adecuados-los-videojuegos-a-los-que-juega-tu-hijo/ en Junio de 2014*
[12] *Imagen obtenida desde http://en.wikipedia.org/wiki/Perfect_Dark_Zero en Junio de 2014*

Sony lanza en 2006 la videoconsola PlayStation 3 y Nintendo responde con la Wii. La primera de ellas incluye la característica de utilizar discos ópticos de H.D. llamados Blue-Ray Disk y un servicio de videojuegos en línea.

Más tarde los tres grandes fabricantes de videoconsolas que quedaban en el mercado Microsoft, Nintendo y Sony (Sega había abandonado este mercado) presentan respectivas mejoras con nuevas versiones como la PSP-GO (Sony) o la DSi XL (Nintendo).

A finales de 2010 y como ejemplo del crecimiento del sector de los videojuegos tenemos que el juego "Call of Duty: Black Ops" llega a obtener los 1.000 millones de dólares de beneficio compitiendo claramente con los obtenidos por las películas de más costo.

El realismo de los juegos hace que sea difícil distinguir su espectacularidad de un film. Así ocurre con videojuegos como "Uncharted 3: Drake's Deception" (para PlayStation 3).

Ya en 2012 llegan al mercado las consolas de la 8ª generación que usan internet como pilar de su funcionamiento y aglutinando las funciones de consolas para juegos y reproductores de películas, series, etc.

Figura 13: Pantalla de Uncharted 3: Drake's Deception [13]

En 2013 Sony lanza la PlayStation 4 con 8 GB de memoria y Nintendo saca la WII U. Microsoft en el mismo año pone a la venta la Xbox One incluyendo el control por medio de voz de sus mandos y controles.

Hasta aquí la densa historia de los videojuegos en ordenadores, máquinas recreativas o videoconsolas. [2]

[13] *Imagen obtenida desde http://www.vandal.net/juegos/ps3/uncharted-3-la-traicion-de-drake/13027 en Junio de 2014*

Pero no debemos acabar este apartado sin hacer mención por su importancia creciente, aunque de historia aún más reciente que la anterior, al desarrollo de los videojuegos en teléfonos móviles. [3] Los videojuegos en este escenario se distinguen mucho de los tradicionales dadas las características propias de estos soportes y se convierten en abiertos y peligrosos competidores. De hecho se convierten en perfectas videoconsolas portátiles que permiten jugar en cualquier momento por llevarlas siempre a mano.

En el principio de la era de los teléfonos móviles (hace unos 10 años) estos aparatos solo servían para hablar y enviar mensajes de texto pero actualmente los teléfonos móviles nos permiten navegar por internet, enviar fotos, conectarnos a redes sociales y evidentemente…jugar, siendo esta última faceta una parte crucial de los Smartphone y las tabletas.

Los juegos en este entorno son por lo general, sencillos y no necesitan mucho tiempo (aunque la complejidad va en aumento), están siempre disponibles y permiten un nuevo tipo de jugador que utiliza este sistema de entretenimiento aunque nunca haya jugado con videoconsolas u ordenadores.

Vamos a ver una historia breve de estos videojuegos:

Fue Nokia el fabricante que decidió añadir a sus móviles el entretenimiento de algunos juegos incluyendo pequeños botones y una pantalla LCD. Los juegos se basaban en los primeros diseños para consolas de los años de la década de 1980. De aquí se pasó a la posibilidad de conectarse a Internet pagando al operador por nuevos niveles.

Los primeros móviles programables se lanzaron en Japón con tecnología Java.

Al principio los videojuegos se programaban en código máquina y se grababan en memoria ROM del propio móvil pero al aparecer los móviles programables se podían grabar nuevos datos en memoria y con un lenguaje como Java y un cable USB se podía cargar de Internet un nuevo juego e incluso aplicaciones para el propio móvil que no tenían por qué ser necesariamente juegos.

Gameloft fue una empresa francesa que llevó a cabo desarrollos de videojuegos para estos soportes con gran éxito.

El primer videojuego de gran popularidad para móviles fue la serpiente “Snake” que pronto se hizo célebre entre poseedores de estos teléfonos por su adictividad a pesar de ser muy simple.

Figura 14: Snake para móviles [14]

Con la evolución de los propios teléfonos móviles en memoria, potencia y lenguajes como Symbian OS o J2ME 2.0, la versatilidad de los videojuegos fue en aumento.

Pero el gran paso se dio en 2007 con la aparición del iPhone. En 2008 se creó la APP Store que permitía subir y vender pequeños programas pequeños programas para dispositivos Apple y que funcionaba como una tienda digital online.

Google, en abierta competencia, lanza el sistema Android y su propia tienda similar al APP Store la llamada Google Play.

En 2009 aparece el juego "Angry Birds", una serie de videojuegos que alcanza una popularidad increíble adaptándose a todo tipo de dispositivos móviles y con pantalla táctil principalmente.

En 2012 llega a acumular más de mil millones de descargas siendo el juego más vendido en la historia para móviles.

Figura 15: Pantalla de Angy Birds [15]

[14] *Imagen obtenida desde http://www.xatakamovil.com/aplicaciones/snake-2k-el-snake-de-los-antiguos-nokia-vuelve-a-tu-Smartphone en Junio de 2014*

[15] *Imagen obtenida desde http://www.esquire.es/actualizacion/824/adictos-a-los-videojuegos-para-moviles en Junio de 2014*

Otros juegos muy populares fueron: la serie “Where’s my…?”, “Apalabrados”, "Cut de Rope” o ”Pou”.

Para darse idea de la magnitud de esta nueva industria, podemos decir que Google Play alcanzaba en octubre de 2012 más de 700.000 aplicaciones y más de 650.000 la APP Store.

Por último añadiremos que el uso generalizado de las tabletas ha permitido ampliar aún más la popularidad de los videojuegos en este soporte y muchos juegos de videoconsolas se comercializan para estos dispositivos siendo, de hecho, grandes competidores.

Como reacción ante este panorama algunos fabricantes de videoconsolas han añadido la posibilidad de poder conectarse al inmenso mercado de aplicaciones como, por ejemplo, es Google Play.

Figura 16: Pantalla de The Walking Dead para Android [16]

[16] *Imagen obtenida desde http://muchotablet.com/juegos-para-tablet/ en Junio de 2014*

2.3 La industria del videojuego en la actualidad

Actualmente la industria de los videojuegos incluye desde el desarrollo de los mismos hasta la venta pasando por la distribución, estudios de mercado etc... Engloba a miles de personas en todo el mundo y a una gran cantidad de disciplinas diferentes.

Como ya se ha expuesto, la industria de los videojuegos tiene un desarrollo vinculado estrechamente con el de la computación, tratamiento de imágenes más reales y cada vez más con el mundo del cine relacionando éxitos cinematográficos con su versión en videojuego generando de hecho ya en la década de 2000 más dinero que el cine y la música juntos. En 2013 llegó a generar 70.000 millones de dólares en todo el mundo y en 2015 se espera que esta cifra alcance los 112.000 millones según estudios llevados a cabo por Gardner.

Al principio era un mundo dominado por chicos y jóvenes pero actualmente incluye a chicas y mujeres de mayor edad e incluso padres y tercera edad y este panorama se modifica aún más en la última década con las aplicaciones a tablets y móviles que ya han alcanzado la madurez con la aparición del teléfono inteligente que hoy alcanza una potencia y una capacidad de procesamiento equiparable a las videoconsolas portátiles.

Ofcom (organismo regulador de las telecomunicaciones) en Gran Bretaña, indica que prácticamente la mitad de los adolescentes tiene un teléfono inteligente y que en más de la mitad de los hogares hay algún tipo de videoconsola. En Estados Unidos, un estudio similar muestra que la actividad del juego es la más popular entre propietarios de teléfonos móviles y tablets, incluso por encima de la navegación por Internet.

El sector de los videojuegos incluye la cadena de valor, actuaciones que antes del despegue de los teléfonos inteligentes y las tabletas incluía (figura 17):

- Los fabricantes de videoconsolas y sus componentes.
- Los creadores del software necesario incluyendo el diseño, el desarrollo y las pruebas.
- Las empresas 'publisher', intermediarias entre el desarrollo y la distribución.
- La distribución que se ubica entre los anteriores y los vendedores al público o detallistas.

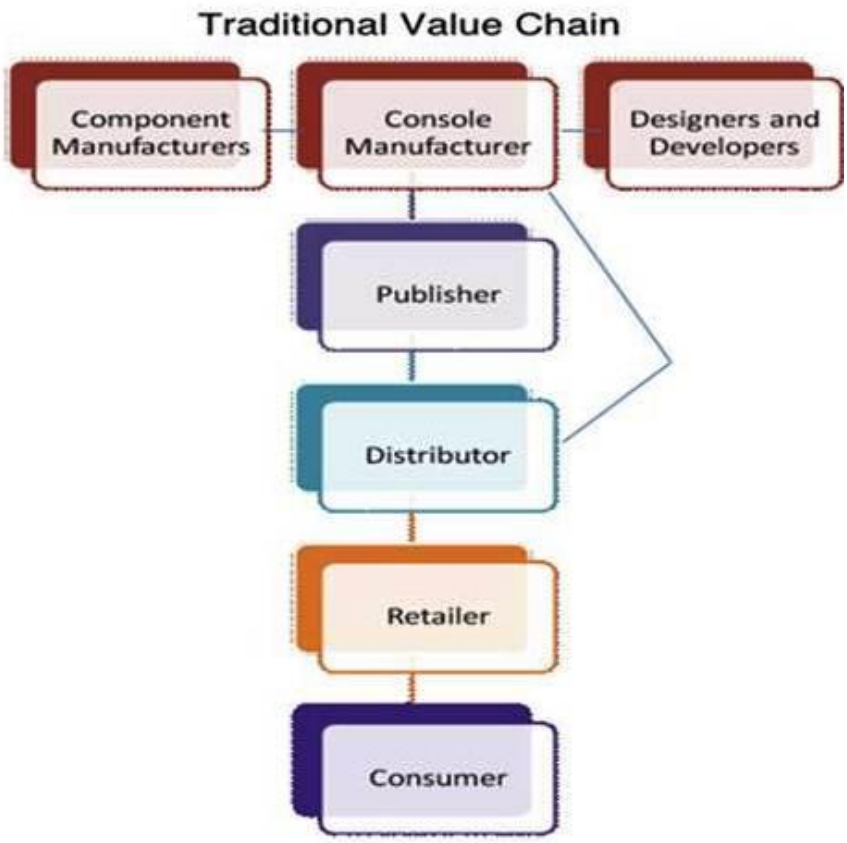


Figura 17: esquema de la cadena de valor tradicional de videojuegos [17]

Pero actualmente esta cadena de valor se ha hecho más compleja constituyendo el principal cambio la irrupción de las aplicaciones para celulares, teléfonos inteligentes y tablets.

Esto da lugar a la aparición de nuevos protagonistas como las cadenas de distribución digital y las empresas creadoras de sistemas operativos para móviles como Appple y los creadores de Android (Google). Esto hace que se añadan a la cadena tradicional nuevos segmentos (figura 18). [4]

Una característica de la industria de los videojuegos es la gran concentración de empresas en el sector tanto en lo que respecta a los fabricantes de videoconsolas como a los desarrolladores.

[17] *Imagen obtenida desde* http://sites.duke.edu/soc142-videogames/global-value-chain/traditional-value-chain/ *en Junio de 2014*

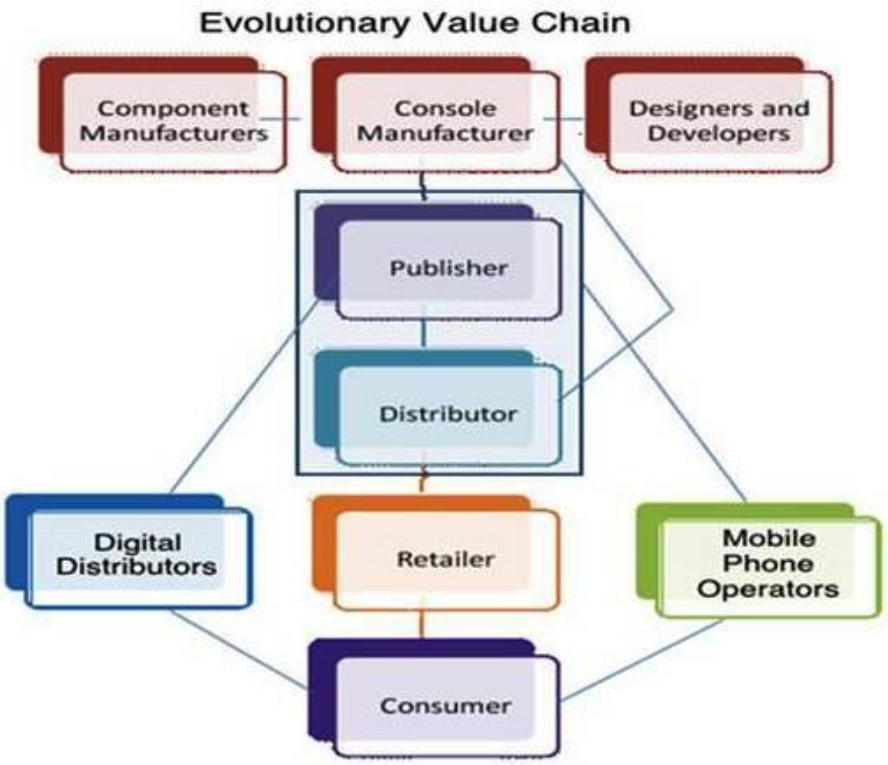


Figura 18: esquema de la nueva cadena de valor videojuegos [18]

De hecho de un estudio realizado al respecto por Metecritic, actualmente no más de 10 empresas abarcan el 70% de la facturación total.

Un lugar más importante en la industria de los videojuegos la ocupan las empresas distribuidoras que son los encargados de distribuir los juegos una vez desarrollados aunque a veces hace también la función de editores y se ocupa en comercializarlos y llevar a cabo las campañas de publicidad y la investigación de los mercados.

En la gráfica elaborada por la ya mencionada Metacritic (figura 19) podemos ver las 10 primeras empresas en volumen de negocio y la variación de su posición en los años 2012 a 2013. Se ha tenido en cuenta en el orden de puntuación general en seis factores: volumen de ventas anuales, número de lanzamientos, puntuación media de críticas, calidad de los productores y extras.

[18] *Imagen obtenida desde* http://sites.duke.edu/soc142-videogames/global-value-chain/evolutionary-value-chain/ *en Junio de 2014*

Posición en 2013	Nombre de distribuidora	Posición en 2012
1	Electronic Arts	4
2	Valve	1
3	Sony	3
4	Nintendo	2
5	Bethesda Softworks	12
6	Rockstar Games	14
7	Ubisoft	6
8	Konami	10
9	Sega	7
10	Activision Blizzard	11

Figura 19: Principales distribuidoras mundiales de vídeojuegos [19]

En lo que respecta a España estamos en un momento que podríamos llamar de valle debido a la recesión económica. Mientras se espera la recuperación se trata fundamentalmente de la creación de una estructura dinámica y sólida que permita un crecimiento similar al de los países de su entorno. Actualmente ocupa la cuarta posición en consumo con relación a Europa pero queda en una posición muy baja en cuanto a producción propia.

En 2013 se generó en nuestro país una cantidad de 762 millones de euros en consumo de videoconsolas y videojuegos.

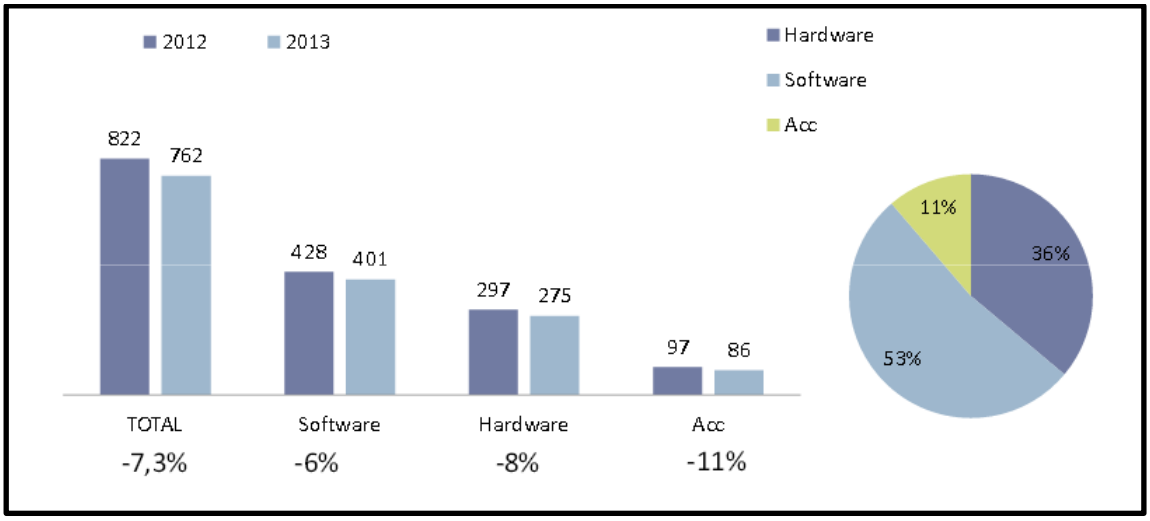


Figura 20: Evolución del mercado de videojuegos en España en 2012-2013 (M€) [20]

[19] *Imagen obtenida desde http://en.wikipedia.org/wiki/Distribuidora_de_videojuegos. en Junio de 2014*
[20] *Imagen obtenida desde* http://www.aevi.org.es/index.php?option=com_mtree&task=att_download&link_id=9&cf_id=30 *en Junio de 2014*

En la figura 20 se puede observar cómo se produce una evolución negativa tanto en software (aplicaciones) como en hardware (equipos) en el consumo con una disminución total del 7,3% por las razones que se han expuesto.

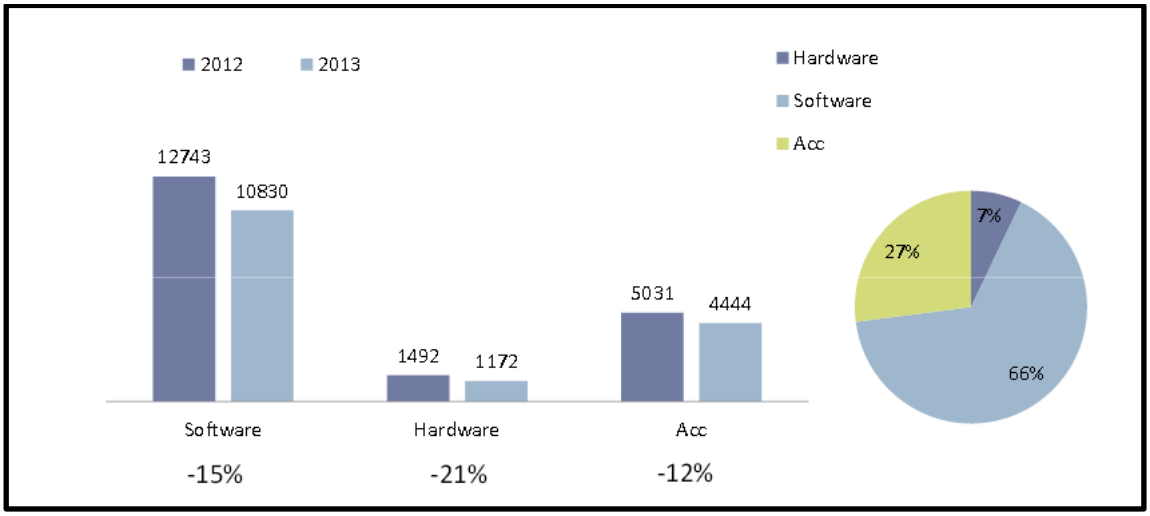


Figura 21: Evolución del mercado de videojuegos en España en 2012-2013 (miles de unidades). [21]

Podemos ver en la figura 21 que para el mismo período anterior hay caída en la distribución de ventas de hasta el 21% de unidades en consolas y periféricos.
A las razones de la crisis económica hemos de añadir un efecto notable producido por la extensión de la piratería en nuestro país tal como informa la ADESE (Asociación Española de Distribuidores y Editores de Software de Entretenimiento). En los juegos para PC es donde se produce el mayor descenso con un 27% en 2013. En este año se vendieron en España un total de 10.830.000 unidades de videojuegos, 1.172.000 consolas y 4.444.000 periféricos.

Como resumen de este análisis vemos que la industria del videojuego tiene un fecundo presente y un más brillante futuro pero nuestro país se enfrenta a fuertes desafíos en los que se incluye la necesidad una buena formación académica y por otro lado es preciso estimular a los empresarios mediante el desarrollo de mejores capacidades que permitan enfrentarse a la dura competencia que existe a nivel mundial.

Hay que tener en cuenta que aunque el mercado esté dominado por las grandes empresas, la situación actual permite que pequeñas y medianas empresas que sepan adaptarse a las nuevas situaciones, puedan encontrar nichos de mercado que les permitan desarrollarse de forma muy notable. [5]

[21] *Imagen obtenida desde* http://www.aevi.org.es/index.php?option=com_mtree&task=att_download&link_id=9&cf_id=30 *en Junio de 2014*

2.4 Los videojuegos educativos

Una vertiente muy importante de los videojuegos es la utilización de los mismos con una finalidad total o parcial de tipo educativo, es decir que el juego trate de motivar al jugador con la enseñanza de un determinado contexto educativo sin, por ello, reducir o eliminar la parte lúdica, siendo ésta la parte utilizada para que sirva de entretenimiento al estudiante logrando el aprendizaje de una manera atractiva.

Algunos de estos juegos están específicamente diseñados para ser utilizados en las propias aulas, o sea, convertirse en herramienta de aprendizaje a través de la cual el profesor inculque conocimientos, hábitos de conducta, aptitudes, etc… de una manera que resulte muy atractiva.

La introducción de los videojuegos en la enseñanza ha sido, hasta ahora, un tema muy discutido porque todavía muchos padres desconfían de esta forma de aprendizaje e incluso consideran que es impropia para estos fines y una pérdida de tiempo.

No obstante se ha demostrado que los videojuegos son bastante efectivos frente a la enseñanza tradicional porque tienen la cualidad de llamar la atención del estudiante, cosa que pocas veces consiguen los métodos tradicionales. De hecho se observa que los niños que utilizan estos métodos aprenden más rápidamente que los que no tienen esta posibilidad pues combinan la diversión y el entretenimiento con el contenido educativo, haciendo más dinámico el proceso de asimilación.

Estas ventajas se hacen aún más patentes si los juegos se desarrollan en grupos incluso si se pueden jugar en red para que los alumnos colaboren entre sí para resolver los problemas planteados en el juego.

En cualquier caso no se pretende utilizar el videojuego para sustituir por completo cualquier otra actividad de la enseñanza tradicional sino, como se ha dicho, es una ayuda muy eficaz para el aprendizaje y que en la mayoría de los casos sirve como complemento. Tanto es así que ya hay editoriales de libros de texto que incluyen en su material estas herramientas como un CD que incluye juegos para que se apliquen los contenidos de las clases.

Los niños en la educación tradicional muchas veces no comprenden el objetivo de su estudio y por ello pierden el interés o tienen que hacer un esfuerzo de voluntad en na actividad que no les atrae porque falta motivación que es precisamente lo que consigue el juego educativo.

Pero también hay quienes advierten de los peligros de esta práctica para la enseñanza achacándoles los efectos nocivos que tradicionalmente se imputan a los videojuegos. Por ejemplo el abuso que se hace de las pantallas con problemas para la vista aunque no deja de ser algo falso para una vista sana. El Profesor Dr. Jorge Alió, Catedrático de Oftalmología y director médico de VISSUM Corporación Oftalmológica, comenta que “las connotaciones negativas de la informática, en general y con relación a la vista, son

un mito". En realidad el efecto es muchas veces beneficioso al estimular la visión e incluso sirve para ser utilizado en algunos casos como el llamado "ojo vago" en algunos escolares.

Otro de los efectos negativos que más se comenta es el problema de la adicción. En este sentido los expertos comentan que la adicción a los videojuegos como problemas de aislamiento, conductas antisociales, bajo rendimiento escolar etc. no tienen como causa el uso de estos métodos sino la falta de responsabilidad, ignorancia o comodidad de padres y educadores para evitar que el niño caiga en semejantes hábitos.

La ineficacia de los videojuegos educativos se produce cuando el estudiante se centra en la parte lúdica haciendo prácticamente caso omiso de la pare educativa con lo que no se obtiene ningún resultado práctico. La mayoría de las veces esto es así por un diseño inapropiado y, en última instancia, es el educador el que debe vigilar el uso adecuado de este procedimiento.

Analizaremos a continuación la penetración en España del videojuego en las aulas y su aceptación y utilización. [5]

Por medio de estudios como el realizado por ADESE (Asociación Española de Distribuidores y Editores de videojuegos) en octubre de 2012 sobre la utilización de los videojuegos en la enseñanza primaria en nuestro país para niños entre 5 y 12 años relacionando la influencia de los videojuegos con el resultado académico y el comportamiento, se llega a conclusiones que son determinantes.

Tanto profesores como padres coinciden en la valoración muy positiva del videojuego como herramienta pedagógica. En el estudio indicado se muestra que más del 75% de profesores de enseñanza primaria tienen una visión muy positiva su utilidad y una gran capacidad como estimulador del aprendizaje. También se muestra que uno de cada 3 profesores de primaria consultados, ya han utilizado esta herramienta alguna vez siendo esta experiencia en un 96% de los casos positiva o muy positiva. Cabe reseñar que son los profesores de menos de 40 años los más inclinados a estos métodos lo que se puede interpretar fácilmente como la típica resistencia del profesorado de mayor edad al cambio de los métodos que llevan practicando tantos años y sobre todo si se trata de métodos en apariencia tan revolucionarios.

En el siguiente gráfico se muestra por medio de porcentajes los motivos para la introducción de los videojuegos en las aulas por parte de los profesores que ya tienen experiencia en su uso.

Figura 22: Porcentaje de respuestas de una muestra de profesores de enseñanza que han utilizado los videojuegos en sus aulas [22]

[22] *Imagen obtenida desde www.adese.es/anuario2012 en Junio de 2014*

2.5 Ejemplos de videojuegos educativos existentes

En este apartado vamos a presentar algunos juegos educativos existentes en el mercado, cuál su temática y su vertiente educativa.

"Aprende con Pipo"

Con este título no se hace referencia a un único juego sino más bien a una colección de videojuegos dedicados a distintas materias con un común personaje que es un niño de primaria y el principal y casi único protagonista.

La colección ha sido desarrollada por la empresa Micronet y su objetivo es la enseñanza de diferentes materias existentes en el sistema educativo que se imparte en España.

Dentro de la gran variedad de títulos podemos reseñar:
"Aprende a leer con Pipo", "Aprende inglés con Pipo", "Aprende música con Pipo", " Viaje en el tiempo con las matemáticas", "Descubre el universo con Pipo", "El cuerpo humano con Pipo", "Geografía con Pipo", "Imagina y crea con Pipo", "Los Animales con Pipo", "Juega con Pipo en la ciudad", "Matemáticas con Pipo", "Mis primeros pasos con Pipo", "Mis primeras palabras en inglés con Pipo" etc…

Figura 23: Pantalla de "Aprende matemáticas con Pipo" [23]

[23] *Imagen obtenida desde* http://rt00149b.eresmas.net/pipo.html *en Junio de 2014*

En la figura anterior se ve una captura del videojuego que se indica que se desarrolla en la China Imperial. En la imagen que se muestra hay varios estandartes con números que se mueven de derecha a izquierda con el fondo de la Gran Muralla China. El estudiante tiene que pulsar en el estandarte que transporta el número que es el resultado correcto de una operación aritmética que aparece en la parte inferior y todo ello en un determinado tiempo máximo.

La dinámica de otros juegos de la misma colección es muy similar a la expuesta para este juego variando el contenido con una serie de preguntas y respuestas, ordenación de números y similares, todo ello acompañado de imágenes y sonido con objeto de llamar la atención.

Este videojuego es el típico ejemplo en el que pesa mucho la parte educativa pero la parte lúdica o de entretenimiento está muy reducida por lo que resulta poco atractivo, es decir el niño apenas si encuentra diversión en su ejecución y además las distintas fases se suceden sin relación entre sí y sin ningún propósito de aventura o habilidad.

A pesar de ello no se le debe quitar su valor instructivo. [8]

"Darfus is dying"

Aunque de momento solo existe la versión inglesa, se trata de un buen ejemplo de equilibrio entre la parte educativa y la parte lúdica y además es interesante su cita porque es una muestra de juego educativo que no trata de obtener conocimientos sino de fomentar pautas de comportamiento y que en este caso en concreto trata de concienciar sobre las dificultades para la supervivencia en zonas en conflicto como las guerras.

El jugador se pone en la piel de un refugiado sudanés que debe sobrevivir utilizando todos los medios a su alcance, incluso obtener agua potable para él y su poblado, burlando las fuerzas que se le oponen o negociando con ellas. También debe decidir la mejor manera de utilizar y distribuir el agua conseguida.

En este caso se comprueba que, además de conseguir el fin educativo que se propone, la dinámica del juego es de un alto interés lúdico y casi equivalente a la de otros juegos no educativos de estrategia o de aventura y acción. [8]

Figura 24: Pantalla de "Darfus is dying" [24]

"Dragon Box "

Es ésta una aplicación bastante original para la enseñanza y aprendizaje de las matemáticas.

Su creador Jean-Baptiste Huynh comenta "Soy profesor de matemáticas, por lo que me encuentro con muchos desafíos, especialmente en Noruega que es donde trabajo, ya que los alumnos no están tan motivados a aprenderlas. Existe una especie de hueco entre las expectativas de los alumnos y lo que ofrecemos como profesores. Tras varios años entendí que en el actual sistema educativo no se innova, porque no se incentiva a los alumnos. No hay nadie ni nada que haga que los alumnos puedan mejorar e innovar en su proceso formativo. Todo el sistema está enfocado sobre los profesores y nunca sobre los alumnos".

Como prueba de su interés se consiguieron, después de haber optimizado su desarrollo, 20.000 descargas en pocos días puesto que quedó disponible en App Store y Google Play aunque también se hizo una versión para PC.

El juego es en sí muy sencillo e intuitivo. Sin embargo su desarrollo implica toda la lógica matemática precisa para aprender álgebra y resolver distintos tipos de ecuaciones. Lo más interesante es que no se requiere ningún conocimiento previo para jugar, solo utilizar la lógica con la motivación que él mismo genera. De hecho este juego fue el considerado como la mejor aplicación de 2012 en Noruega y premiado con la llave de oro de este país.

En 2013, Dragón Box fue seleccionado por los Premios Internacionales de Juegos para Móviles (IMGA) como el mejor "juego de aprendizaje" en competición con los ya seleccionados como los mejores en este género.

[24] *Imagen obtenida desde* http://rt00149b.eresmas.net/darfus.html *en Junio de 2014*

Figura 25: Una carátula del juego Dragon Box [25]

Está traducido a doce idiomas y dirigido especialmente a estudiantes entre 10 y 13 años sin tener conocimientos previos de álgebra pero próximos a ingresar en la enseñanza secundaria.

Como prueba de su eficacia se puede indicar que en pruebas realizadas con estudiantes de incluso edades tan tempranas como de 5 y 6 años que todavía no saben operaciones aritméticas sencillas como sumar y restar, los resultados han sido también satisfactorios. [9]

Hakitzu

Es un ejemplo de videojuego de distribución gratuita para IOs y pensado para la enseñanza de programación en JavaScript, es decir un tipo de enseñanza que se aleja de las tradicionales en cuanto no suelen ser materias escolares.
Se lanzó al mercado por "Kuato Studios".

Su temática cosiste en enfrentar robots luchando entre sí. El usuario va codificando instrucciones, cada vez de mayor complejidad, necesarias para el movimiento de los robots y el uso de las armas de defensa y ataque.

Previamente al lanzamiento del videojuego se llevó a cabo por la empresa un estudio muy completo en los centros escolares de Estados Unidos y Gran Bretaña con objeto de probarlo con grupos de estudiantes y comprobar su eficacia en el aprendizaje y su interés lúdico.

[25] *Imagen obtenida desde* http://www.dragonboxapp.com/ *en Junio de 2014*

Con todo ello resulta un producto atrayente y entretenido para el aprendizaje de la programación. [10]

Figura 26: Pantalla de Hakitzu [26]

Para terminar mencionaremos otros videojuegos educativos también interesantes sin extendernos tanto en su contenido. [10]

Minecraft

Ejemplo de videojuego no pensado en principio para tener una vertiente educativa pero que posteriormente se hicieron versiones para ser utilizadas en las aulas.

Proyecto Kokori

Su área educativa es la biología.

Pequeñas naves "nanobot" se introducen en distintas partes de los organismos vivos, incluso a nivel celular, para ser comprendidos y estudiados.

Cap Odyssey

Desarrollado por la empresa francesa KTM para el Ministerio de Agricultura francés.

Permite introducir al jugador el mundo agrícola (economía, precios, mejora de la producción, riesgos del clima, plagas, etc...) y con objeto de motivar a los jóvenes en un mundo cada vez más alejado del interés de los mismos.

[26] *Imagen obtenida desde* http://mashable.com/2013/03/27/hakitzu/ *en Junio de 2014*

2.6 Software existente para la creación y desarrollo de videojuegos

A continuación, vamos a hablar sobre algunas de las herramientas existentes más conocidas para la creación de videojuegos. En concreto, vamos a poner como ejemplo 5 programas que nos van a permitir iniciarnos en el desarrollo de los videojuegos, sin necesidad de tener grandes conocimientos sobre la materia.

Stencyl:

Esta aplicación implementada para sistemas **Windows y Mac OS** introduce el sistema conocido como Drag & Drop (arrastrar y soltar), utilizada para el desarrollo y creación de videojuegos, que permite convertir en algo distraído e incluso divertido lo que de otra manera sería bastante más difícil y complicado, lo que nos permite el diseño de juegos de creación propia para PC con solo utilizar elementos que se arrastran y sueltan. Así, se crean escenarios, se determinan acción, escenas, etc… dando la forma y acabado muy personales.

No obstante esto no quiere decir que la creación de un juego no sea una tarea compleja, incluso con el manejo de Stencyl, por lo que se recomienda estudiar los correspondientes tutoriales y emplear el tiempo suficiente de estudio para practicar el comportamiento de las distintas opciones que tiene el programa. Con todo ello se puede resumir que podemos obtener un manejo satisfactorio en un tiempo bastante breve.

Existen 3 versiones del programa: uno gratuito básico, uno de paga básico y un tercero de estudio para iOS y Android. [11]

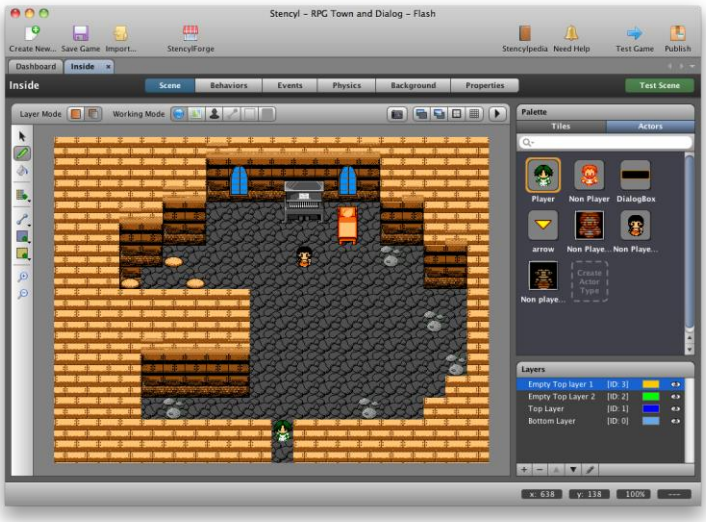


Figura 27: Entorno de trabajo de Stencyl [27]

[27] *Imagen obtenida desde* http://blog.stencyl.com/wp-content/uploads/2012/04/Screen-shot-2012-04-24-at-10.07.43-PM.png *en Junio de 2014*

Game Develop:

Se trata de un software libre para diseño de videojuegos 2D, que permite la creación de juegos para Linux y Windows. Contiene un interfaz *drag & drop* muy sencillo de usar pero a la vez potente para ser utilizado por usuarios avanzados.

Existen a disposición del usuario tutoriales para su aprendizaje. Como permite la creación de videojuegos en HTML5, se puede exportar a Facebook.

Algunas de sus características son:

- Permite la creación de videojuegos sin apenas conocimientos de programación, sin por ello dejar de estar personalizado según nuestros propios criterios.

- La interfaz de usuario es muy intuitiva y sencilla, creando ambientes para los personajes de manera visual y sin necesidad de utilización de código, incluyendo efectos de sonido y banda sonora, con los correspondientes entornos y ambientes que deseamos.

- Game Develop tiene su propio compilador que le capacita para exportar los juegos a aplicaciones en el entorno Windows, Linux y Mac OS.

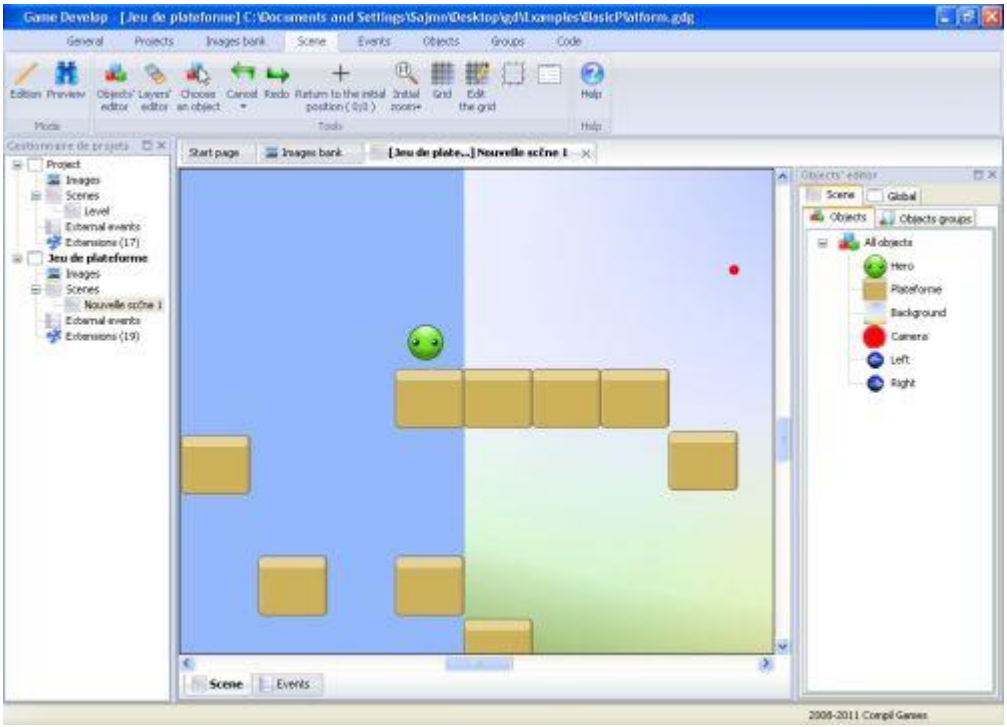


Figura 28: Entorno de trabajo de Game Develop [28]

[28] *Imagen obtenida desde* http://www.techsupportalert.com/files/images/Game%20Develop-500.jpg *en Junio de 2014*

Entidad 3D

Herramienta, que como su nombre indica, sirve para desarrollar videojuegos 3D para PCs con prácticamente todas las versiones de Windows, permitiendo la creación de juegos similares a Quake, Half-Life o Counter Strike, incluyendo la posibilidad de añadir algún desarrollo o argumentos propios que haga al jugador llevar acabo alguna tarea u objetivo, aparte de matar enemigos, como puede ser: recoger objetos para ser utilizados o entregados a otros personajes a cambio de alguna ayuda o abrir puertas con estos objetos, etc… También se pueden añadir armas defensivas u ofensivas, disponiendo de 4 armas diferentes y configurables.

Permite, por tanto, la creación de muy diversos escenarios y tipos de juegos como pueden ser: arcade, aventura, plataformas, FPS, TPS, RPG… etc y con modalidades en primera o en tercera persona.

Y todo esto se puede realizar sin necesidad de programar en ningún lenguaje de programación y pudiendo distribuir el juego que hagamos libremente de forma gratuita. [13]

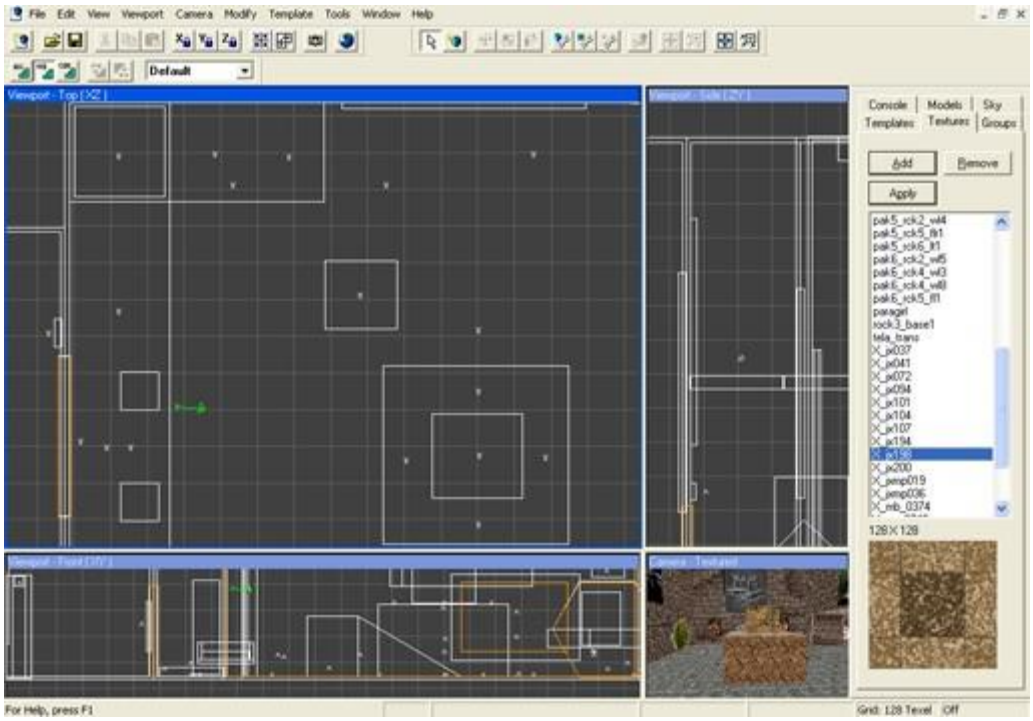


Figura 29: Entorno de trabajo de Entidad 3D [29]

[29] *Imagen obtenida desde* http://www.entidad-3d.com/tutorial_entidad_3d/mejoras30/avance_we2.jpg *en Junio de 2014*

Construct 2

Con esta herramienta podemos diseñar juegos en formato HTML5, que tampoco requieren conocimientos específicos de programación, pues está basada, al igual que algunos de los ya citadas con el sistema "drag and drop", es decir, en un entorno exclusivamente visual.

Contiene un motor para producir efectos físicos para añadir a nuestros juegos dependiendo de la habilidad y la imaginación del diseñador, el atractivo y la calidad del producto final, pudiendo conseguirse muy buenos resultados.

Posee 3 niveles de licencia: Free, Standard y Business. La primera, que lógicamente tiene ciertas limitaciones, se puede descargar libremente por el tiempo que deseemos pero impide comercializar el juego. La segunda tiene un cierto coste (unos 80 dolares), ofrece más características y además nos permite comercializar, con ciertas restricciones, el juego diseñado. La tercera tiene un mayor coste (unos 360 dólares), con aún más posibilidades de diseño y no tiene restricciones en su comercialización. [14]

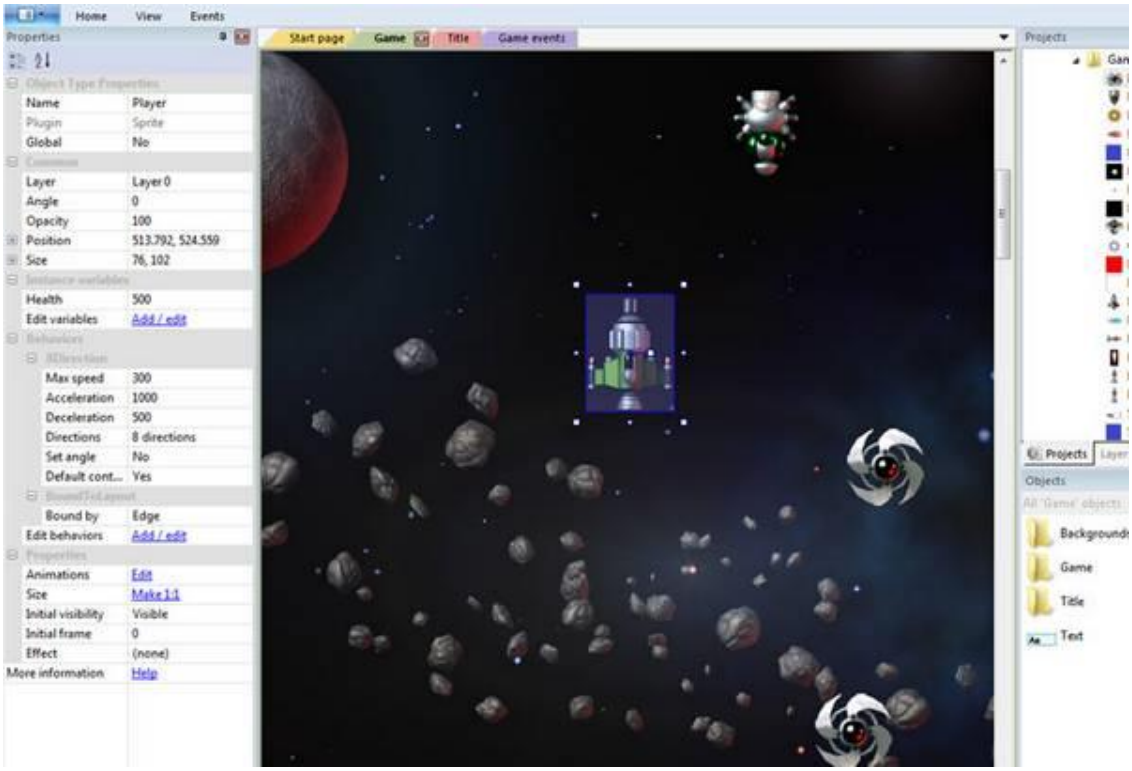


Figura 30: Entorno de trabajo de Game Develop [30]

Para más información: https://www.scirra.com

[30] *Imagen obtenida desde* http://geeksroom.com/wp-content/uploads/2012/04/construct-2.jpg *en Junio de 2014*

Kodu Game Lab

Es una herramienta con un entorno de desarrollo integrado o IDE (de sus siglas en inglés) creado por FUSE Labs de Microsoft y nos permite crear videojuegos para Windows y para la plataforma Xbox 360.

Una característica interesante es que fue diseñada para que los niños pudieran acceder a ella siendo por tanto de interfaz sumamente sencilla y creando entornos y personajes a partir de elementos que están previamente configurados, permitiendo crear juegos para ejecutarse en entornos 3D.

Al igual que la mayoría ya citados, no necesita ningún conocimiento de programación, siendo su manejo totalmente visual basado en Logo (lenguaje de programación educativo) y otros proyectos más recientes como AgentSheets, Squeak y Alice.

Existen versiones para descargar como Indie Game Xbox 360 y para PC en versión beta, disponibles para todos por ser descargable desde una página web.

Presenta diferencias notables respecto de otras herramientas en los siguientes aspectos:

- Utilizan elementos visuales a través de un dispositivo de juego, evitando código tipificado.
- Los programas se ejecutan en entorno 3D, en lugar de pantalla de mapas de bit o 2D.

Kodu Game Lab, incluso se utiliza como herramienta educativa en centros de aprendizaje y escuelas y se encuentra disponible de forma gratuita, existiendo también una versión para descargar en la plataforma Xbox 360 como un Indie Game. [15]

Figura 31: Pantalla principal de Game Develop [31]

[31] *Imagen obtenida desde* http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-95-13/2260.image005.jpg *en Junio de 2014*

2.7 Unity 3D

2.7.1 ¿Qué es Unity 3D?

Esta herramienta sirve para el desarrollo de videojuegos en varias plataformas utilizando un editor y scripting para su creación, dando a dichos juegos un final de corte profesional.

Unity 3D presenta varias versiones para su uso: una gratuita y otra profesional de pago. Cada una tiene sus limitaciones y sus ventajas. Lógicamente, la última versión es la más completa aunque ello conlleva un coste que no está al alcance de todos los diseñadores, sobre todo si estos están empezando a utilizar dicha herramienta.

Como ya se ha expuesto, su uso sirve para muchas plataformas como PC, Mac, Nintendo Wii y iPhone y tiene un motor gráfico 3D para Mac y PC, sirviendo no solo para crear juegos sino también aplicaciones interactivas, animaciones 3D, etc.
También pueden desarrollarse juegos para la web con el plugin Unity web player.

El contenido del videojuego se construye desde el editor, que es de tipo visual muy completo y útil en su manejo ya que por medio de unos cuantos clicks podemos importar modelos 3D, sonidos, texturas, etc… para trabajar con ellos.

Por medio de la herramienta de desarrollo MonoDevelop, podemos crear scripts en JavaScript, C# y un dialecto de Python llamado Boo, extendiendo así las posibilidades de dicho editor utilizado las APIs del que está provisto junto a los correspondientes tutoriales en su página web.

Los videojuegos diseñados en Unity se estructuran en escenas. Cada escena puede ser cualquier parte del videojuego desde el menú de inicio hasta un nivel del juego, tomando el diseñador la elección correspondiente, ya que cada escena es un como un lienzo en blanco para dibujar cada parte del juego.

También se incluye en el motor un editor de terrenos donde poder crear uno concreto sobre el que representar su forma y características usando herramientas de tipo visual, pintar, cubrir de hierba, texturizar, poner árboles, etc… importados desde aplicaciones como 3DS Max o Maya.

Por otra parte, si se necesitan más recursos, podemos dentro de la propia aplicación, acceder a una tienda similar a la APP Store, llamada Asset Store, con multitud de recursos de pago o gratuitos, pudiendo incluso ampliar los recursos de la herramienta mediante plugins [16].

2.7.2 Criterios para la decisión de Unity 3D

En este apartado vamos a mencionar cuales son las causas por las cuales hemos decidido utilizar Unity 3D como herramienta de desarrollo para nuestro juego.
Las ventajas que ofrece esta herramienta en el diseño de videojuegos se mencionan a continuación:

- Está disponible en **versión gratuita y profesional**, siendo esta última la que permite un mayor número de opciones. Unity dispone de dos licencias principales: Unity y Unity Pro, siendo gratuita la primera y con un determinado coste la segunda. Originalmente costaba alrededor de 200 USD (ahora $1500). La versión Pro tiene ciertas características añadidas. Algunos ejemplos son render a textura, determinación de cara oculta, iluminación global y efectos de post-procesamiento. La versión gratuita indica a su comienzo del juego una pantalla de bienvenida (en juegos independientes) y una marca de agua (en los juegos web) que no se puede desactivar o personalizar.

  Unity y Unity Pro incluyen en el entorno de desarrollo ejemplos de proyectos tutoriales, ayuda a través de foros, wiki, y las actualizaciones futuras de la misma versión principal. En nuestro caso, evidentemente para la realización del proyecto se ha elegido la versión gratuita, ya que permite hacer todo lo que queremos, sin coste alguno. Lo positivo es que Unity Technologies ha puesto a disposición de los desarrolladores independientes **sus herramientas de desarrollo bajo Unity de manera gratuita**, y además para todas las principales plataformas móviles: iOS, Android, Windows Phone 8 y Blackberry 10.

  Por otro lado, hay que destacar que el problema que tenía Unity era que para poder comercializar el juego para la plataforma iOS o Android mediante la App Store, debías pagar una licencia de uso que ascendía a 800 dólares, valor que muchos desarrolladores no podían permitirse ya que seguramente no recaudarían ni la mitad de ese importe con su creación. Recientemente, Unity cambió esta política y ahora, en el caso en que se quisiera publicar nuestro proyecto en la Play Store, la licencia para publicar juegos basados en Unity pasa a ser totalmente gratuita. Solamente será necesario pagar la licencia de uso todos aquellos desarrolladores o empresas que ingresen más de 100.000$ con su desarrollo.

- Permite **diseñar juegos para variedad de plataformas** como PC, Mac y Web, así como para dispositivos móviles.

  Unity sigue el lema de **"crear una vez, Implementar en todas partes"** soportando el despliegue de múltiples plataformas. Unity le permite desarrollar para todas las plataformas y poder cambiar entre ellas con una sola herramienta. En un mismo proyecto se puede tener control, pudiendo implementarlo a todas las plataformas como teléfonos móviles, webs, escritorios y consolas. Unity permite también controlar algunas otras propiedades como la especificación de compresión de texturas y ajustes de resolución para cada plataforma para la compatibilidad del

juego. Esto permite que un solo archivo de alta resolución funcione para todos los destinos. De esta forma, Unity permite a los desarrolladores centrar sus esfuerzos en la creación de videojuegos, en vez de tener que enfrentarse además al duro trabajo de fondo que requiere adaptarlo a diferentes plataformas.

Concretamente, Unity permite exportar juegos a las plataformas Windows, OS X, Linux, Xbox 360, PlayStation 3, Playstation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone. Además, gracias al Plug-In Web de Unity, se pueden desarrollar juegos de navegador, para Windows y Mac.

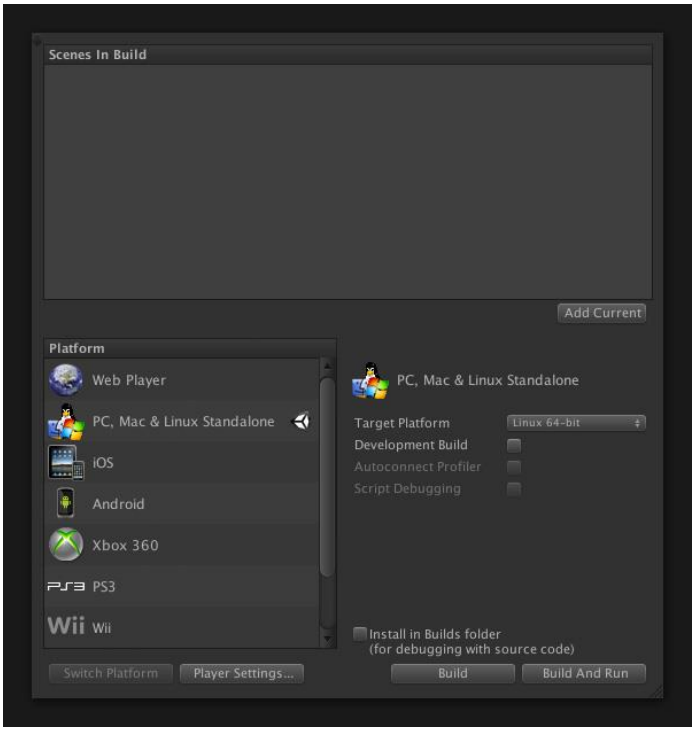


Figura 32: Ventana de "build settings" de Unity 3D [32]

- Otra característica importante es que puede **utilizarse en los distintos sistemas operativos** disponibles actualmente, no solamente Windows. Unity está disponible en su instalación como plataforma de desarrollo para Windows, OS X y Linux. Además, la **facilidad en el uso del editor gráfico** y la instalación de plugins lo convierte en una herramienta muy eficiente.

  Contiene una App Store llamada Unity Asset Store mediante la cual pueden añadirse recursos adicionales al videojuego. En noviembre de 2010 fue lanzado este recurso disponible en el editor. Todos los usuarios de Unity pueden acceder a la colección de más de 4.400 paquetes de Assets (y sigue en continuo crecimiento) en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música y efectos de sonido, tutoriales, proyectos, scripts, extensiones para el editor y servicios en línea.

[32] *Imagen obtenida desde* http://unity3d.com/profiles/unity3d/themes/unity/images/unity/multiplatform/build-dialog-big.jpg *en Junio de 2014*

2.7.3 Juegos desarrollados con Unity 3D

A continuación mostramos una lista de los juegos desarrollados con este motor y el año en el que fueron publicados: [17]

- *Dead Frontier* (2008)
- *Three Kingdoms Online* (2008)
- *Cartoon Network Universe: FusionFall* (2009)
- *Max & the Magic Marker* (2010)
- Thomas Was Alone (2010)
- *Battlestar Galactica Online* (2011)
- *Family Guy Online* (2011)
- I Am Playr (2011)
- *Rochard* (2011)
- *Nihilumbra* (2012)
- *Dead Lab* (2012)
- *No Heroes* (2013)
- *Shadowgun* (2011)
- *Triple Town* (2011)
- *Bad Piggies* (2012)
- *Dead Trigger* (2012)
- *Endless Space* (2012)
- *Escape Plan* (2012)
- *Guns of Icarus Online* (2012)
- *MechWarrior Tactics* (2012)
- *Prime World* (2012)
- *Slender: The Eight Pages* (2012)
- *Temple Run* (2012)
- *Among the Sleep* (2013)
- *Game of Thrones: Seven Kingdoms* (2013)
- Hearthstone: Heroes of Warcraft (2013)
- *Interstellar Marines* (2013)
- *République* (2013)
- *Wasteland 2* (2013)
- *Project Eternity* (2014)
- Rust (Alpha 2013, TBD)
- *Castle Story* (TBD)

- War for the Overworld (2013)
- Dreamfall Chapters: The Longest Journey (2014)
- *Kerbal Space Program* (TBD)
- *Shadowrun Online* (TBD)
- Kartrider Dash (Facebook)
- Red Crucible 2 (Facebook)
- *Torment: Tides of Numenera*
- Unearthed: Trail of Ibn Battuta (TBD)
- Slender: The Arrival (2013)

Capítulo 3: Especificación de requisitos

3.1 Recopilando ideas para un videojuego

En este apartado, vamos a estudiar detenidamente los elementos que queremos incluir en el mismo y los requisitos a cumplir. Es decir, la fase de "concepción" o aquella en la que se pretende encontrar los elementos clave que puedan gustar a la gente y que consigan tener éxito.

Para ello se han llevado a cabo las siguientes ideas:

- Separar por completo su parte educativa de su parte lúdica centrando los esfuerzos en buscar un juego excitante, entretenido y que enganche.

- Al añadir la parte didáctica al juego, apenas nos demos cuenta de que este ingrediente existe considerándolo como un elemento de dificultad añadido y que permite poner los conocimientos a prueba del jugador sin quitarle la diversión y el entretenimiento con el que fue concebido.

- Que el jugador que se encuentra delante del juego no lo vea aburrido y soso. Además, que el jugador pueda evaluarse de esta forma acerca de una materia y cuantas más respuestas acierte, más facilidad tendrá a la hora de ganar, pero no tiene por qué convertirse en el factor imprescindible para obtener la victoria. Simplemente es uno más.

- Que a medida que el jugador reintente de nuevo la partida, las preguntas puedan llegar a repetirse, y esta es la mejor forma de que el usuario memorice y aprenda la respuesta de una determinada pregunta, a través de la reiteración. Es decir, buscar la idea preliminar del juego que queremos y más tarde añadir el ingrediente pedagógico.

Una vez recogidos estos puntos, se hacen propios para nuestro proyecto:

- El juego tiene que mantener toda la atención del jugador, concentrando su atención en un objetivo muy claro. Esto es necesario para que el jugador no se aburra y pierda el interés. Tiene que mantenerle tenso y en suspense. No vale cometer errores o eso repercutirá en el resultado.

- El juego tiene que tener comportamientos aleatorios. Para darle realismo se necesita que los enemigos puedan elegir libremente el camino que trazarán hasta llegar a su destino. Esto permite que cada partida o nivel sea totalmente distinto al anterior.

- El juego tiene que ofrecerle al jugador la libertad, no solo de movimiento (por ejemplo para evitar a los enemigos) sino para crear una estrategia de actuación y vencerlos. Esto dará lugar a que el juego tenga un mundo más abierto a diferentes posibilidades y combinaciones de actuación para lograr nuestro objetivo.

- El juego tiene que tener un grado de dificultad creciente. Los usuarios tienen que sentirse atrapados por el mismo y sentir que pueden ir superando los primeros niveles con cierta facilidad hasta que la dificultad de los siguientes va poniendo más a prueba su habilidad y destreza.

- Tiene que ser un juego que induzca a seguir jugando. Cada vez que el jugador entre en la pantalla principal, aparecerá la puntuación máxima alcanzada, haciendo que éste se motive más para intentar superarse.

- Las sesiones deben ser relativamente cortas y que se puedan jugar en cualquier lugar o momento. Por tanto que sea atractivo y divierta en poco tiempo.

A partir de estos elementos comenzamos a idear cual sería el juego que protagonizaría nuestro proyecto.

3.2 Plantillas de especificación de requisitos:

Vamos a rellenar lo que serían las plantillas para especificación de requisitos más importantes de nuestro proyecto.
Los requisitos de usuario seguirán el siguiente formato:

Identificador: RU-TXXX			
Nombre	*Nombre del requisito.*		
Prioridad	Alta / Media / Baja	**Fuente**	Cliente / Otros
Necesidad	Esencial / Deseable / Opcional		
Claridad	Alta / Media / Baja	**Verificabilidad**	Alta / Media / Baja
Estabilidad	*Estabilidad del requisito.*		
Descripción	*Pequeña descripción del requisito*.		

Formato de requisitos de usuario

Donde:

- Identificador: identifica unívocamente un requisito. Se descompone en los siguientes campos:
  - RU: requisito de usuario.
  - T: identifica el tipo de requisito que puede ser F (Funcional) o NF (No Funcional).
  - XXX: numeración consecutiva.
- Nombre: título del requisito, para saber de qué trata el requisito sin necesidad de leerse la descripción.
- Prioridad: orden en el desarrollo de los distintos módulos del sistema.
- Fuente: quién o qué es el origen del requisito.
- Necesidad: indica cuán deseable es la inclusión de un requisito en el sistema final.
- Claridad: indica si un requisito de usuario tiene una, y sólo una, interpretación. La claridad implica la falta de ambigüedad.
- Verificabilidad: indica si es posible comprobar fehacientemente que el requisito se ha incorporado en el diseño.
- Estabilidad: establece si el requisito podrá modificarse en un futuro.
- Descripción: explicación del requisito.

Requisitos funcionales:
Describen el comportamiento del sistema. Por ejemplo, cuál es una determinada acción posible en el sistema y cómo interacciona el usuario con dicho sistema y qué respuestas obtiene de él. A continuación se exponen los requisitos funcionales obtenidos del videojuego a desarrollar:

Identificador: RU-F001			
Nombre	*Características generales y objetivos*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema*		
Descripción	El juego deberá estar orientado a todos los públicos, especialmente al público más joven, siendo un juego atractivo, adictivo, fácil de usar, intuitivo y donde haya que cumplir un objetivo bien definido: impedir que los monstruos lleguen a la aldea.		

Tabla 1. Requisito de usuario RU-F001

Identificador: RU-F002			
Nombre	*Pantalla de inicio y estadísticas*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Podrá modificarse ciertas características del mismo en un futuro*		
Descripción	El juego deberá mostrar una pantalla de inicio que indique una pequeña presentación en forma de animación de fondo, con el título del juego, así como información acerca de las partidas anteriores: puntuación y oleada máxima alcanzada.		

Tabla 2. Requisito de usuario RU-F002

Identificador: RU-F003			
Nombre	*Iniciar partida*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema*		
Descripción	El sistema deberá permitir iniciar una nueva partida siempre que el usuario lo desee. Para ello deberá deslizar el dedo sobre la pantalla, una vez que se encuentre en la pantalla principal.		

Tabla 3. Requisito de usuario RU-F003

Identificador: RU-F004			
Nombre	*Menú principal*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema*		
Descripción	El juego deberá contener en la pantalla principal un menú que permita al usuario pueda acceder a otras opciones adicionales como son: • Tutorial: el sistema muestra un pantallazo con anotaciones que describen cada uno de los elementos que se mostrarán en pantalla cuando el jugador inicie la partida • Créditos: el sistema muestra una imagen indicando el autor del proyecto y la fecha en el que ha sido desarrollado • Salir: permite salirse del juego		

Tabla 4. Requisito de usuario RU-F004

Identificador: RU-F005			
Nombre	*Ciclo del juego*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Media
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego deberá tener un principio y un final, siendo posible que el jugador pueda terminar el juego con éxito. En caso de no tener éxito se le devolverá a la pantalla de inicio		

Tabla 5. Requisito de usuario RU-F005

Identificador: RU-F005			
Nombre	*Fases del juego*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial / Deseable / Opcional		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Tiene que estar basado en sesiones de juego relativamente cortas y que se puedan jugar en cualquier lugar o momento. El juego se dividirá en sucesivas rondas u oleadas que conformarán los niveles por los que irá pasando desde su comienzo hasta su final. En total hay un número de 10 niveles. Si la partida finaliza, podremos reintentarlo, empezando desde el principio en el nivel 1.		

Tabla 6. Requisito de usuario RU-F006

Identificador: RU-F007			
Nombre	Enemigos		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	En cada ronda, irán apareciendo un número de monstruos que se irán moviendo desde la zona más alejada del mapa (norte) e irán atravesando el laberinto por diferentes caminos hasta llegar a la aldea (sur). Existirán 5 monstruos distintos, cada uno con sus propias características como velocidad, salud máxima, etc…		

Tabla 7. Requisito de usuario RU-F007

Identificador: RU-F008			
Nombre	*Comportamiento aleatorio de los enemigos*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego tiene que tener comportamientos aleatorios. Para darle realismo se necesita que los enemigos puedan elegir libremente el camino que trazarán hasta llegar a su destino. Esto permite que cada partida o nivel sea totalmente distinto al anterior.		

Tabla 8. Requisito de usuario RU-F008

Identificador: RU-F009			
Nombre	*Comportamiento sistemático de los enemigos*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	A lo largo del camino, si los monstruos se topan con un objeto defensivo como son las vallas o las torretas, o bien llegan a la aldea, estos comenzarán a atacar hasta destruir o acabar con su objetivo. Si se encuentran a nuestro personaje a poca distancia, estos intentarán seguirnos hasta lograr alcanzarnos. Si no, los monstruos desistirán y seguirán su camino hacia la aldea.		

Tabla 9. Requisito de usuario RU-F009

Identificador: RU-F010			
Nombre	Dificultad		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego tiene que tener un grado de dificultad creciente. Cuando acabamos con los monstruos de una de las rondas, se pasará al siguiente nivel donde aparecerán nuevos monstruos más difíciles o un número mayor de estos, así como la combinación de diferentes tipos. Esto tendrá lugar de forma sucesiva, cada vez que consigamos pasar de nivel.		

Tabla 10. Requisito de usuario RU-F010

Identificador: RU-F011			
Nombre	*Parte educativa*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego deberá tener una parte educativa en la que se probarán los conocimientos del usuario. El jugador deberá acertar diversas cuestiones que se le formularán para obtener las herramientas y lograr vencer a los monstruos. Estas herramientas estarán formadas por torretas o vallas que se podrán colocar por el camino.		

Tabla 11. Requisito de usuario RU-F011

Identificador: RU-F012			
Nombre	*Información de la partida*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego aportará la información necesaria al jugador mientras está jugando, pudiendo visualizar los datos más importantes como son: el nivel, la puntuación y los monstruos restantes para pasar a la siguiente ronda así como la vida de nuestro personaje, la vida de la aldea y el número de torretas o vallas que disponemos para colocar.		

Tabla 12. Requisito de usuario RU-F012

Identificador: RU-F013			
Nombre	*Menú de partida*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego deberá disponer de un botón que permita desplegar un menú una vez iniciada la partida que permita. Al pulsar dicho botón, el juego quedará pausado a la espera de que el jugador tome una decisión. Las opciones que se mostrarán serán las siguientes: • Continuar: Permitirá reanudar la partida • Reiniciar: La partida se reiniciará, volviendo a iniciarse la partida desde el nivel 1 • Salir: Permitirá salirse directamente del juego		

Tabla 13. Requisito de usuario RU-F013

Identificador: RU-F014			
Nombre	*Final del juego*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Cuando el jugador alcance el final del juego con éxito, se mostrará una pantalla de fin. Cuando el jugador haya fracasado, bien porque se le hayan acabado sus vidas o bien porque hayas destruido la aldea, el juego mostrará una pantalla de Game Over, seguido de la puntuación alcanzada.		

Tabla 14. Requisito de usuario RU-F014

Identificador: RU-F015			
Nombre	Puntuación y oleada máxima alcanzada		
Prioridad	Media	**Fuente**	Desarrollador
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Al momento de llegar a la pantalla de Game Over, el juego comparará la puntuación alcanzada y comprobará si se ha superado la máxima, en cuyo caso se mostrará el mensaje NUEVO RECORD seguido de la puntuación. En ese momento, la puntuación y la oleada máxima serán guardadas para que el jugador pueda verlo en la pantalla de inicio.		

Tabla 15. Requisito de usuario RU-F015

Identificador: RU-F016			
Nombre	Modificación y edición de las preguntas		
Prioridad	Alta	**Fuente**	Docente u otros
Necesidad	Opcional		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Modificable en un futuro*		
Descripción	El docente podrá modificar el fichero XML de preguntas sin necesidad de tener conocimientos sobre el código interno del juego. Tan solo necesitará modificar el fichero “plantilla.xml” almacenado en la carpeta “resources” de los Assets del proyecto y volver a compilar el juego, haciendo uso de Unity 3D. Posteriormente podrá publicar el juego nuevamente en Play Store.		

Tabla 16. Requisito de usuario RU-F016

Identificador: RU-F017			
Nombre	*Desplazamiento del personaje*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Nada más iniciarse la partida, el usuario podrá manejar el desplazamiento del personaje a través del Joystick situado en la parte de inferior derecha de la pantalla.		

Tabla 17. Requisito de usuario RU-F017

Identificador: RU-F018			
Nombre	*Cuándo se lanzarán las preguntas*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Al pasar nuestro personaje por un cubo de color, el juego lanzará una pregunta aleatoria contenida en el fichero XML con tres posibles respuestas que el usuario deberá acertar para conseguir una torreta o una valla. Una vez contestada, el cubo desaparecerá del mapa.		

Tabla 18. Requisito de usuario RU-F018

Identificador: RU-F019			
Nombre	*Controles para defensa*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El juego permitirá que el usuario coloque las vallas o torretas que disponga con los botones situados en la parte inferior izquierda, fruto de haber acertado las preguntas. Si el jugador no dispone de objetos, no podrá colocarlos aunque pulse los botones. Debajo de los botones, se visualizarán los contadores que nos indican cuantas torretas o vayas disponemos para poder colocar. Cuando el usuario recoja o coloque uno de estos elementos, los contadores mostrados debajo de los botones se incrementarán o decrementarán según corresponda.		

Tabla 19. Requisito de usuario RU-F019

Identificador: RU-F020			
Nombre	*Lugares donde podrán situarse nuestros elementos defensivos*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Las vallas podrán situarse únicamente en aquellas zonas del suelo donde se dibuja una línea clara. Las torretas podrán ser situadas en cualquier lugar donde el usuario crea conveniente para atacar a los monstruos.		

Tabla 20. Requisito de usuario RU-F020

Identificador: RU-F021			
Nombre	*Comportamiento de la aldea en el juego*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Cuando los monstruos lleguen a la aldea, empezará a disminuir la barra de vida de la misma hasta que alcance el final, en cuyo caso habremos fracasado y saldrá la pantalla de Game Over		

Tabla 21. Requisito de usuario RU-F021

Identificador: RU-F022			
Nombre	*Vidas del personaje*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El personaje, nada más iniciarse la partida, dispondrá de 3 vidas, representado arriba a la derecha con 3 cruces con corazón. Cuando el personaje es atacado, se eliminará una vida. Si es atacado sucesivas veces, el usuario dispone de unos segundos para huir antes de que le quiten la siguiente vida. El personaje puede coger hasta un máximo de 5 vidas que irá encontrando a lo largo del mapa. Cuando el personaje dispone solo de una vida, si vuelve a ser atacado, se acabará la partida mostrándose una pantalla de GameOver.		

Tabla 22. Requisito de usuario RU-F022

Requisitos no funcionales:
Describen características internas del software. Por ejemplo, tiempos de ejecución máximos, o necesidad de seguir algún estándar.

Identificador: RU-NF001			
Nombre	*Lenguaje de programación: JavaScript*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Baja
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Se refiere al lenguaje de programación utilizado en los scripts para desarrollo del proyecto en Unity 3D. Dentro de los 3 posibles lenguajes que admite Unity, se ha escogido JavaScript por su similitud con el lenguaje Java estudiado en la carrera.		

Tabla 23. Requisito de usuario RU-NF001

Identificador: RU-NF002			
Nombre	*Compatibilidad con dispositivos*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Media
Estabilidad	*Durante toda la vida del sistema*		
Descripción	El sistema deberá ser compatible con diferentes dispositivos táctiles (móviles o tablets) con sistema operativo Android 2.3.1 o superior		

Tabla 24. Requisito de usuario RU-NF002

Identificador: RU-NF003			
Nombre	*Compatibilidad con resoluciones de pantalla*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Media
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El sistema ha sido ideado para un dispositivo táctil con una resolución óptima de 480x800. No se ha probado para dispositivos con otras resoluciones		

Tabla 25. Requisito de usuario RU-NF003

Identificador: RU-NF004			
Nombre	*Acceso a internet*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	Será necesario que el usuario disponga de acceso a internet en el dispositivo móvil o táctil, así como una cuenta en Google para tener acceso al Play Store de donde se descargará el juego		

Tabla 26. Requisito de usuario RU-NF004

Identificador: RU-NF005			
Nombre	*Idioma*		
Prioridad	Alta	**Fuente**	Desarrollador
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Modificable en un futuro*		
Descripción	El idioma que presentará el interfaz, así como los diferentes datos qué proporcionará el videojuego será el español.		

Tabla 27. Requisito de usuario RU-NF005

Identificador: RU-NF006			
Nombre	Estructuración de las preguntas del fichero XML		
Prioridad	Media	**Fuente**	Desarrollador
Necesidad	Deseable		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	*Durante toda la vida del sistema.*		
Descripción	El docente o persona encargada de editar el fichero XML con las cuestiones, deberá hacerlo siempre siguiendo la siguiente estructura: <cuestionario> <pregunta1> <enunciado> Enunciado de la pregunta </enunciado> <respuesta1> respuesta 1</respuesta1> <respuesta2> respuesta 2 </respuesta2> <respuesta3> respuesta 3 </respuesta3> <correcta> respuesta correcta </correcta> <pregunta1> <pregunta2> …… …… </pregunta2> </cuestionario>		

Tabla 028. Requisito de usuario RU-NF006

3.3 Casos de uso:

A continuación se mostrará en este apartado los casos de uso y la secuencia de eventos típica de esta aplicación por parte del docente y por parte del alumno o jugador. Seguidamente, se visualizará a través de un esquema, las diferentes acciones que podrá realizar el docente antes de que el usuario se descargue la aplicación, así como las acciones disponibles para el usuario una vez ponga en ejecución el juego.

- **Nombre del caso de uso:** Contain the Monsters

- **Actores:** Usuario y docente

- **Propósito:** Entretener y pasar un buen rato aprendiendo y poniendo a prueba nuestros conocimientos y habilidades. Los conocimientos de los que nos queramos evaluar podrán ser considerados y añadidos por un docente.

- **Descripción:** El usuario, una vez descargado el juego, puede iniciar una partida y tendrá como objetivo impedir que los monstruos lleguen a la aldea. Para ello, se servirá de torretas automáticas y vallas que tendrá que colocar en el camino. Para recoger estas herramientas, el usuario deberá acertar las preguntas que se le proponen al recoger un cubo amarillo o rojo.

- **Secuencia de eventos típica:**

    1. El docente elige qué preguntas se formularán al usuario, editando un fichero XML que formará parte de los archivos del juego.

    2. Una vez editado, el docente procederá a compilar el juego en un único archivo ejecutable con extensión APK para dispositivos Android.

    3. El docente procederá a subirlo a algún servidor como PlayStore o lo dejará disponible en algún lugar para que el usuario pueda descargarlo o transferirlo a su dispositivo móvil.

    4. El usuario, una vez que se lo ha descargado, accede al juego donde lo primero que se mostrará será la pantalla principal con la puntuación y la oleada máxima alcanzada.

    5. Si lo desea, lo primero que hará será visualizar el tutorial que se encuentra en el menú principal para entender un poco los elementos qué se mostrarán en pantalla una vez inicie la partida. Para ello pulsará sobre el botón “menú” y elegirá la opción “tutorial”

    6. Si el usuario lo desea, también puede visualizar el creador de la aplicación a través de la opción “créditos” que se encuentra en el menú de la pantalla principal.

7. Si el usuario desea salir del juego deberá elegir la opción “salir” del menú principal.

8. Para iniciar la partida, el usuario deslizará el dedo sobre la pantalla.

9. El usuario inicia la partida, y empieza a moverse por el laberinto, recogiendo los cubos rojo y amarillo para ir acumulando las torretas y vallas que posteriormente deberá colocar.

10. A medida que el usuario recorre el laberinto, los monstruos van atravesando el mapa por diferentes caminos para llegar a la aldea.

11. Cada vez que el usuario recoge un cubo, se le formulará una pregunta con 3 posibles respuestas de las cuales tendrá que seleccionar la correcta. En caso de acertar, dependiendo del color del cubo, el jugador recibirá una torreta en caso de ser un cubo rojo, o una valla en caso de ser un cubo amarillo.

12. El jugador deberá impedir la llegada de los monstruos colocando las torretas o las vallas recogidas. Las vallas impedirán temporalmente su entrada y las torretas acabarán con ellos.

13. Si el usuario acaba con el número de monstruos de dicho nivel, pasará a un siguiente nivel donde la dificultad irá incrementándose progresivamente hasta llegar al final del juego, en cuyo caso el jugador habrá ganado la partida.

14. Si los monstruos alcanzan la aldea, estos empezarán a atacarla, disminuyendo la barra de vida de la misma. Al llegar la barra al final, habremos perdido y habrá terminado la partida.

15. Si los monstruos alcanzan a nuestro personaje, las vidas irán disminuyendo hasta que solo nos quede una, en cuyo caso si los monstruos nos vuelven a alcanzar, la partida terminará y habremos perdido.
    Para impedir esto, podremos recoger vidas que encontraremos por el camino, acumulando hasta un máximo de 5.

16. Una vez que el usuario se encuentra jugando, puede pausar el juego pulsando sobre el botón menú, en cuyo caso se le mostrarán 3 opciones posibles.

17. Si el jugador desea continuar, pulsará el botón “continuar” para que el juego vuelva a retomarse desde el último punto

18. Si el jugador desea reiniciar la partida, pulsará el botón “reiniciar” para volver al nivel 1 y empezar de nuevo.

19. Si el usuario decide salir del juego, deberá pulsar el botón “salir” del menú de la pantalla de juego.

20. Una vez que acabe la partida, bien porque ganemos o bien porque hayamos perdido, el juego mostrará una pantalla de fin del juego con la puntuación y nos devolverá al menú principal.

Casos de uso: Diagrama UML:

A continuación se muestra el diagrama UML con los casos de uso del docente y del alumno. En el caso del alumno, distinguiremos entre los casos de uso desde el menú de la pantalla principal y los casos de uso desde el menú de la pantalla de juego.

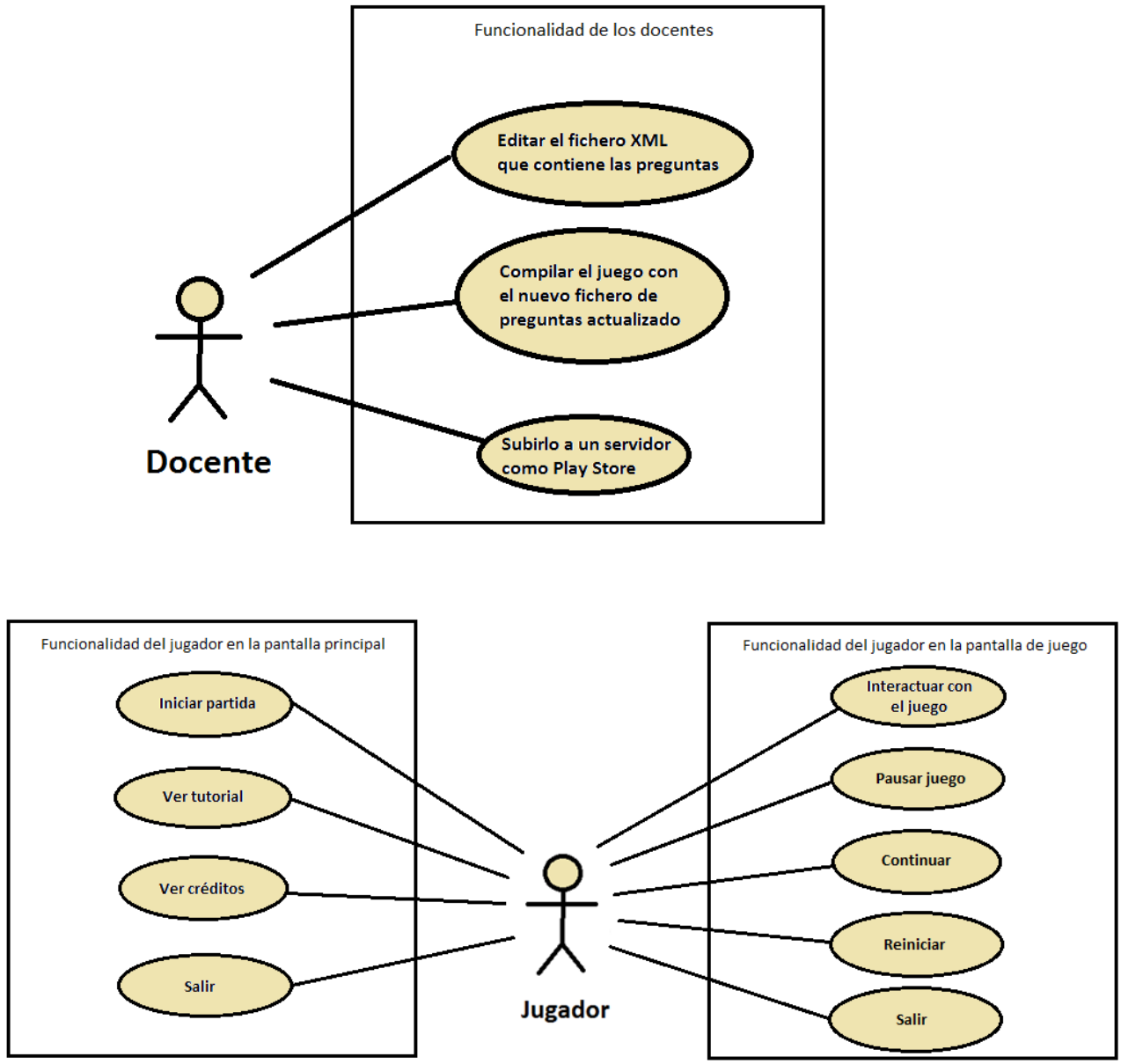


Figura 33: Diagrama de Casos de Uso

3.4 Elementos y estrategias introducidas en "Contain the Monsters"

De la suma de todos los ingredientes anteriores nace "Contain the Monsters".

Contain the Monsters podría definirse en el género **Tower defense** o **TD.**
Vamos a hacer una pequeña introducción de en qué se basan este tipo de juegos según Wikipedia [18]:

Se trata de un subgénero de los videojuegos de estrategia en tiempo real y cuyo objetivo es "*lograr que las unidades enemigas no lleguen a cruzar el mapa*. *Para lograrlo se deben construir torres que las atacan al pasar. Tanto los enemigos como las torres tienen diferentes habilidades y costes. Al eliminar una unidad enemiga se reciben puntos o dinero que debe ser utilizado para construir o mejorar torres. La elección apropiada de torres y su ubicación es la estrategia esencial del género. Muchos juegos empiezan con un laberinto que será atravesado por los enemigos, al cual se le construyen torres a los lados del camino. Otros comienzan como un mapa libre para que el usuario coloque torres. En este último caso, las unidades enemigas deben contar con cierta inteligencia artificial para encontrar el camino hasta el final del escenario".*

Según estas bases, se introducen en el juego los siguientes elementos:

- Un laberinto como escenario principal para poder definir diferentes caminos que podrían ser atravesados por los enemigos, así como una forma de retrasar el paso de estos y aumentar las posibilidades de contenerlos y que no lleguen a su objetivo.

- Todos los monstruos parten de la parte más alejada del mapa, del norte, y nuestro personaje inician su andadura desde el sur donde se halla la aldea solitaria, hasta los diferentes puntos del mapa para acabar con los enemigos.

Figura 34: Ejemplo del modo Supervivencia del exitoso juego: Gear of War Judgment [33]

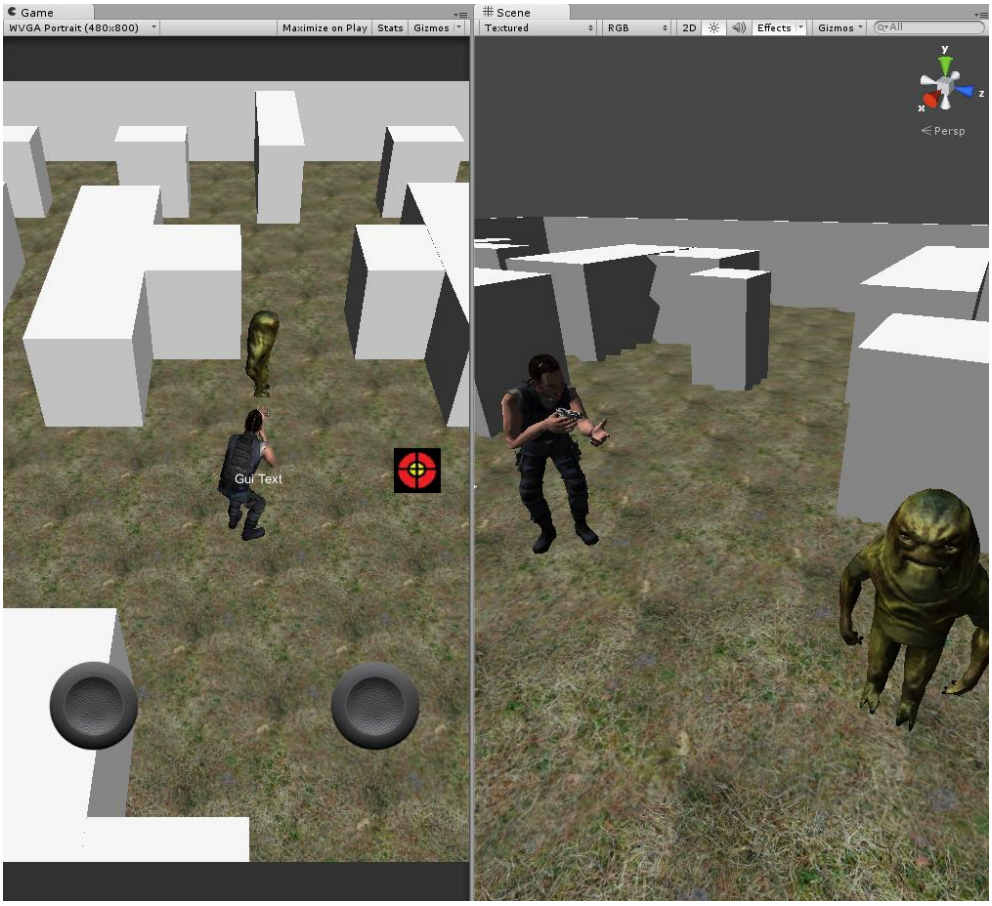


Figura 35: Captura de una versión en desarrollo del proyecto

- Se suprime la idea de que el personaje pueda disparar directamente a los monstruos, como puede verse en la figura superior, por creación de torretas como forma de acabar con los enemigos, dando la libertad al jugador de colocarlas en aquellos lugares donde puedan serle de ayuda mientras recorre el mapa en busca de más herramientas que retuviesen la llegada de enemigos.

Figura 36: Modelo 3D de las torretas usadas en el proyecto

[33] *Imagen obtenida desde* http://img4.meristation.com/files/imagenes/general/vlcsnap-2013-03-17-15h52m26s80.png *en Junio de 2014*

- Empleo de vallas de verjas como formas de obstaculizar por un tiempo la llegada de monstruos. Estas tienen un límite de duración, basado en los golpes y la fuerza que tengan los monstruos. Las vallas permiten al personaje seguir recorriendo el mapa mientras mantiene a los monstruos entretenidos, destrozando la verja. Esto da como resultado un gran dinamismo al juego, pues nos sirve para centrarnos en diferentes puntos para lograr nuestro objetivo.

  Combinar las vallas con las torretas y situarlas correctamente será necesario para alcanzar la victoria.

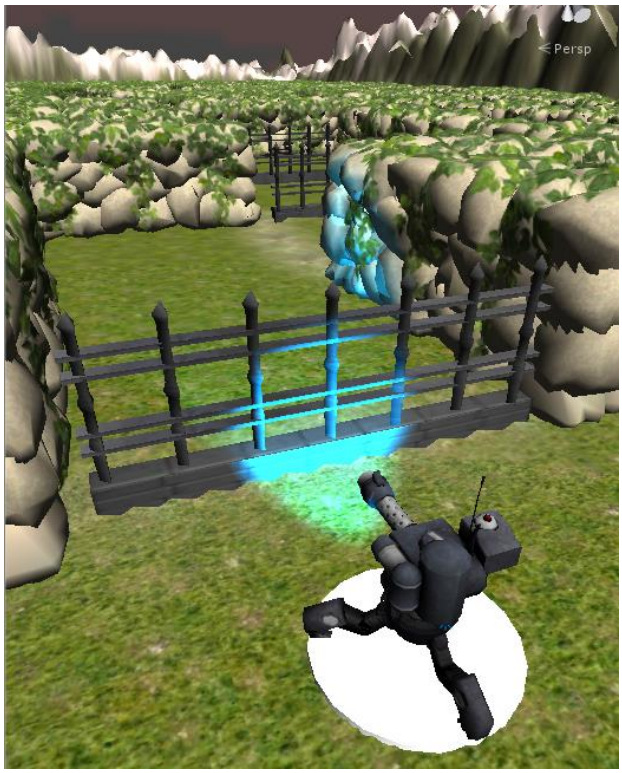


Figura 37: Combinación de torretas con vallas en el juego

- Para introducir en el juego la parte educativa, cada torreta se corresponde con un cubo rojo en el mapa, así como cada valla que recojamos se corresponde con un cubo amarillo. Al pasar por un cubo y recogerlo se lanzará una pregunta que pondrá al jugador a prueba. Si el jugador consigue responder satisfactoriamente se le concederá su objeto defensivo, dependiendo del color del cubo. Si no es acertada, el cubo desaparecerá y habrá perdido el objeto.

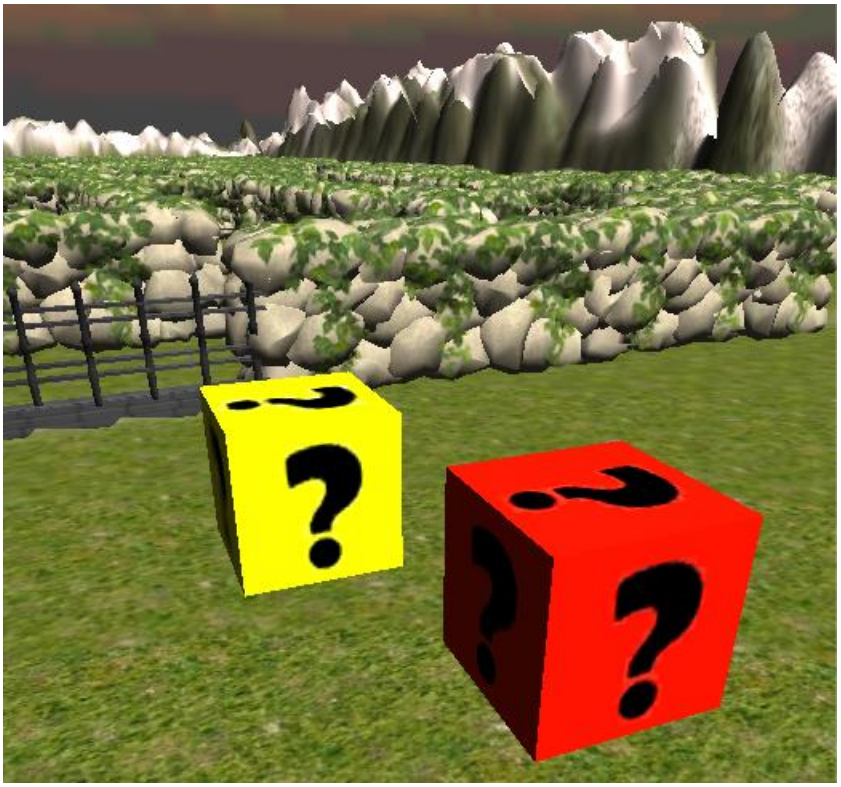


Figura 38: Cubos de colores que contienen las preguntas

Así pues, responder correctamente es esencial para obtener la victoria pero también es importante plantear una estrategia, pues desde que comenzamos la partida hasta que la terminamos el número de cubos se mantendrá constante.

Esto quiere decir que, para ganar, el jugador tendrá que:

- Responder correctamente al mayor número de preguntas posible para tener más garantías de defensa, así como una mayor puntuación.

- Colocar estratégicamente los objetos para conseguir acabar con el mayor número de enemigos y alcanzar un nivel más alto.

- Utilizar el mínimo número de recursos, esto es, tener en cuenta de que los cubos así como las vidas del personaje no se regeneran en cada nivel. Está pensado para que el personaje use un número máximo de vallas y torretas a lo largo del juego y que con éstas trata de obtener la victoria.

Capítulo 4: Diseño e implementación

4.1 Diseño del proyecto y elementos que conforman la Jerarquía.

Como se ha visto anteriormente, Unity es un motor gráfico 3D que viene empaquetado como una herramienta para crear juegos, aplicaciones interactivas, visualizaciones y animaciones en 3D y en tiempo real.

Los juegos creados en Unity son estructurados en escenas. Una escena puede ser vista como un nivel dentro del juego, pero también podría contener el menú de opciones, el menú de pausa, el menú principal o cualquier otra cosa.

Por algunas razones que explicaremos más adelante, nuestro proyecto se ha estructurado en una única escena llamada "**EscenaDelJuego.unity**" y donde se almacena todos los objetos que intervendrán en el juego, llamados GameObject.
Todo y todos los objetos dentro del juego son GameObjects. Pueden ser muy simples o extremadamente complejos, visibles o invisibles, pero si existe en el juego es un GameObject.

Para hacer una vista rápida a cada uno de los elementos y cómo se encuentran ordenados, pasamos a exponer la Vista de Jerarquía.

La Vista de Jerarquía contiene todos los objetos que intervienen en la escena actual. Sirve para navegar por ellos y de este modo entender su funcionamiento en el juego y como se encuentran organizados.

A su vez, estos objetos o GameObjects vamos a dividirlos en 2 grupos: los objetos llamados prefabs y los objetos permanentes en la vista de la jerarquía. A continuación vamos a explicar en qué consisten estos dos grupos y cuáles son los objetos que los conforman.

4.1.1 Objetos permanentes en la Vista de la Jerarquía

En primer lugar se analizan aquellos objetos que permanecen siempre presentes en la Vista de la Jerarquía. Se caracterizan porque se cargan e instancian en el momento en que ejecutamos el juego y no se destruyen durante el transcurso de este. Se les programa para que cumplan una función a través de los diferentes scripts que llevan asociados, se activan o desactivan según se requiera, pero su instancia siempre permanece y sus elementos siempre aparecen representados en la Vista de la Jerarquía.

Vamos a indicar una breve descripción de cada uno de ellos:

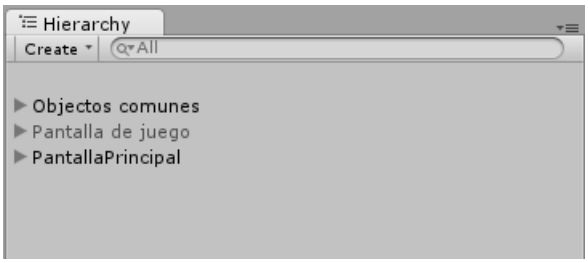


Figura 39: Captura de la ventana de jerarquía de Unity

- **Objetos comunes:** se trata de aquellos objetos que intervienen tanto en la representación de la pantalla principal, como de la pantalla de juego.

- **Pantalla de juego:** aglutina los objetos que intervienen en la partida cuando el usuario está jugando

- **Pantalla principal:** es la pantalla de inicio y presentación del juego. La pantalla principal se visualizará al principio de ejecutar el juego y cuando el usuario termine la partida o pierda.

A continuación pasaremos a detallar de qué elementos se componen los objetos principales indicados y profundizaremos en cuál es su función.

4.1.1.1 Pantalla principal

En todo juego que se precie, encontraremos una pantalla o menú principal donde se nos presentará el título del juego así como una breve introducción del mismo en forma de imágenes o animación. Además, se añadirán diferentes opciones a seleccionar como comenzar una nueva partida, guardarla, retornarla… etc.

Se ha querido construir una pantalla principal sencilla, donde se muestre arriba el título y abajo los resultados más altos alcanzados por el jugador.

Abajo se anuncia la frase “deslice el dedo para comenzar”, que invita al jugador a deslizar el dedo sobre cualquier punto de la pantalla para iniciar la partida. A la derecha, hemos querido representar a nuestro personaje principal, seguido de una torreta al fondo a la izquierda que estará vigilando, moviéndose de un lado a otro.
A la izquierda, más próximo al personaje se encuentra un cartel donde se visualizará la palabra “menú” que nos permitirá, pulsando sobre él, acceder a opciones adicionales.

Finalmente se muestran al fondo del todo, las casas que deberemos proteger y estar atentos para que no sean alcanzadas por los monstruos.

Figura 40: Captura de la pantalla principal

A continuación, analizamos cada uno de los objetos que intervienen en la jerarquía de la pantalla principal.

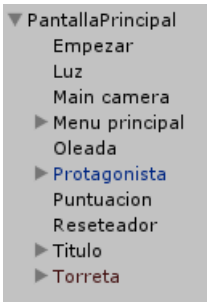


Figura 41: Jerarquía de "Pantalla Principal"

- **Empezar:** se trata de un objeto tipo Text Mesh, al cual le hemos aplicado una animación en bucle. Dicha animación se basará en un movimiento cíclico, basado en aumentar y disminuir su escala para dar el efecto resaltador deseado al texto: "DESLICE EL DEDO PARA COMENZAR"

- **Luz:** a la escena se le aplicará una luz direccional para iluminar el escenario que mostramos.

- **Main camera:** se trata de la cámara principal que muestra nuestra pantalla principal

- **Menu principal:** Consiste en el objeto encargado de mostrar al jugador las diferentes opciones adicionales que tiene. Gráficamente, se representa en la pantalla principal a través del dibujo de un cartel clavado en el suelo que lleva escrita la palabra "Menú". Para entender su funcionamiento, vamos a indicar los elementos de los que está compuesto:

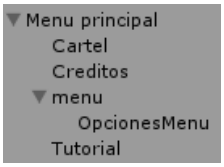


Figura 42: Jerarquía de "Menu principal"

  - ✓ **Cartel**: Se trata de un GUITextute con el dibujo de un cartel que lleva escrito la palabra "Menú". Lleva asociado el script "**touchMenu.js**" cuya finalidad es activar o desactivar el menú que se despliega en pantalla al pulsarse dicho icono.
    El dibujo fue obtenido de la página:
    http://www.colorearjunior.com/dibujos-para-pintar-de-letrero-de-madera-en-la-monta%C3%B1a-se%C3%B1al_10114.html
    con sus respectivas modificaciones

  - ✓ **menu**: Tiene como hijo el objeto "**OpcionesMenu**" y es el encargado de desplegar la pantalla de menú cuando se pulsa el cartel.
    La imagen del pergamino ha sido obtenida de:
    http://www.imagui.com/a/pergamino-hd-TMdXoyo8j
    OpcionesMenu lleva asociado el script "**menuPrincipal.js**", el cual contiene las instrucciones que se ejecutarán al pulsarse una de las 3 opciones mostradas en el menú.

Las 3 opciones serán:

- **Tutorial**: al pulsar sobre esta opción, se mostrará en pantalla una imagen (objeto GUITexture "**Tutorial**") a modo de guía, donde se describirá brevemente cada uno de los elementos que aparecerán en el juego y cómo utilizarlos.

- **Créditos**: se mostrará una pantalla una imagen (objeto GUITexture "**Creditos**") con el autor del juego y la fecha en la que ha sido desarrollado

- **Salir**: permite al jugador salirse del juego

Figura 43: Captura del menú de la pantalla principal

- ✓ **Tutorial**: Se trata de un GUITexture con la imagen del tutorial y que lleva asociado el script "**touchTutorial.js**" que contiene las instrucciones necesarias para mostrar o no la imagen según corresponda.

- ✓ **Créditos**: Se trata de un GUITexture con la imagen de los créditos y que lleva asociado el script "**touchCreditos.js**" que contiene las instrucciones necesarias para mostrar o no la imagen según corresponda.
  La imagen del pergamino ha sido obtenida de: http://www.elmundomagicodeamanda.com/t6441-invitacion-pergamino-con-molde con sus respectivas modificaciones.

- **Oleada:** se trata de otro Text Mesh cuya función es indicar al jugador la oleada máxima alcanzada en partidas anteriores. Lleva asociado un script llamado "**CargarOleadaMaxima.js**" encargada de cargar el nivel más alto que hemos alcanzado.

- **Puntuación:** es el HighScore de la pantalla y mostrará la puntuación máxima alcanzada. Está basado en un Text Mesh al cual se le asocia un script llamado "**CargarHighScore.js**" y cuya función es cargar la puntuación máxima guardada en el móvil de partidas anteriores.

- **Protagonista:** se trata de nuestro personaje principal elegido, el cual es un niño, y cuya animación elegida para esta escena será la animación "idle", que es la animación por defecto que se suele asignar a los personajes 3D cuando no están haciendo ninguna acción. El modelo 3D ha sido obtenido gratuitamente a través del Asset Store, tiene como nombre "Cartoon Boy and Girl" y se encuentra en la sección 3D Models > Characters > Toons. Publicador: Vu studios

- **Reseteador:** es el encargado de que, cuando el jugador pulse cualquier punto de la pantalla y la partida se inicie, todos los valores o resultados anteriores sean reseteados, así como la cantidad de torretas y vallas acumuladas por el personaje. Este objeto lleva asociados 2 scripts:

  - "**DeslizarParaEmpezar.js**", cuya función es invocar todas las funciones que permiten resetear los valores desde cero, cuando el usuario desliza el dedo sobre la pantalla.

  - "**Resetador.js**", se trata del script que será llamado por el script anterior y cuyas funciones permitirán fundamentalmente: destruir los objetos colocados por el personaje en la pantalla anterior, desactivar o activar los objetos que hayan sido modificados por el personaje, reiniciar la barra de vida de la aldea, reiniciar el "**controlador**", reiniciar el número de vidas del

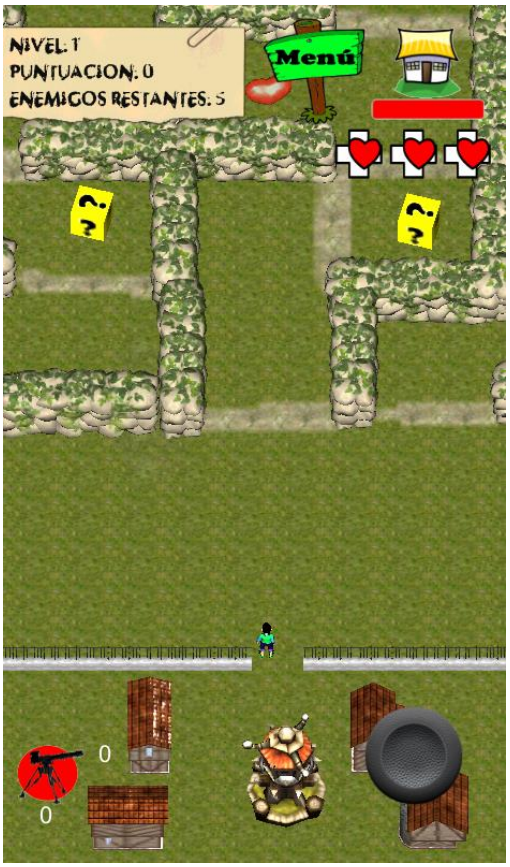
Player, reiniciar valores referentes a la partida actual como son el número de vallas y torretas que tenemos, así como reiniciar el nivel y la puntuación alcanzada

- **Título:** es el título del juego, formado por 2 Text Mesh donde se lee: Contain the Monsters

- **Torreta:** se trata del objeto torreta, el cual utilizaremos más adelante en el juego para defendernos de los monstruos. En la pantalla principal, tan solo nos interesa mostrarla girando en si misma de un lado a otro. En cuanto a su funcionamiento y los scripts que la componen, lo explicaremos más detalladamente en otro apartado.
  El modelo 3D ha sido obtenido del Asset Store en la sección 3D Models > Props > Weapons

4.1.1.2 Pantalla de juego

El jugador deslizará el dedo sobre la pantalla y se iniciará el juego. En este momento, los objetos hijo que formaban parte del objeto "**PantallaPrincipal**" se desactivarán, así como su "**main camera**" y se dará paso a la activación del objeto "**PantallaJuego**", que será el que aglutine todos los objetos propios que intervendrán en la partida.



A primera vista, cuando comenzamos la partida podemos observar a nuestro personaje desde arriba, delante de la aldea que tendrá que proteger.
Arriba a la izquierda se mostrará en forma de trozo de papel agarrado como con un clip, una pantalla de información acerca de cuál es el nivel en el que estamos jugando, la puntuación que llevamos que dependerá de la cantidad de preguntas acertadas, así como de los monstruos abatidos y finalmente los enemigos restantes que nos quedan para terminar la ronda.
Más a la derecha tenemos un dibujo de una casa que representa la aldea que tratamos de proteger. Debajo de la misma observamos una barra de color rojo que simboliza el estado actual de la aldea y su nivel de salud o destrucción con respecto a los monstruos. A medida que estos la alcancen y empiecen a atacarla, la barra irá disminuyendo hasta llegar el final.

Figura 44: Captura de la pantalla de juego

Más abajo tendremos el número de vidas de nuestro personaje, o veces que éste podrá aguantar un ataque de los monstruos. En principio comenzamos con 3 y podremos coger un máximo de 5 a través de los corazones que aparecerán girando en el mapa. Cuando únicamente nos quede una vida y seamos atacados o tocados por un monstruo, la partida finalizará también.

Abajo a la derecha se situará el Joystick con el que el jugador moverá a nuestro protagonista y a la izquierda tendremos los 2 botones que nos servirán para colocar las torretas o las vallas por el mapa. Debajo de estos botones podremos conocer cuál es el número de torretas o vallas que tenemos a nuestra disposición para colocar, y que se irán sumando según vayamos recogiendo los cubos del mapa y vayamos acertando las preguntas que se nos formulan.

A continuación vamos a ir analizando los elementos específicos de la jerarquía que intervienen en la **pantalla de juego**:

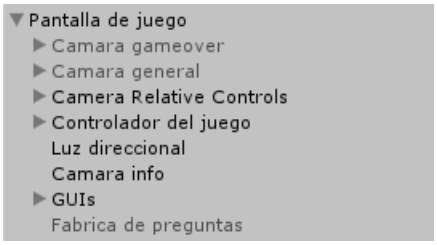


Figura 45: Jerarquía de "Pantalla de juego"

- **Camara gameover:** Se trata de la cámara que se activará al perder la partida, bien porque los monstruos acaben con la casa o bien porque nuestro personaje pierda todas las vidas.

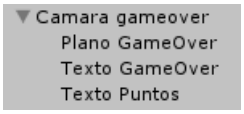


Figura 46: Jerarquía de "Cámara gameover"

  Está formado por un fondo semitransparente de color blanco el cual llamamos "**Plano GameOver**" y el texto "**Game Over**" que saldrá superpuesto al fondo, seguido de otro texto que nos indicará la puntuación alcanzada. En caso de que consigamos superar la puntuación máxima, este texto mostrará "NUEVO RECORD**:**", seguido de la puntuación.

- **Camara general:** La cámara general es aquella que se activará cuando pulsemos sobre el icono con el dibujo de la casa que se encuentra arriba a la derecha.

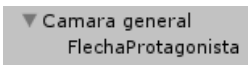


Figura 47: Jerarquía de "Cámara general"

Tiene como finalidad poder ver de un plumazo la posición en la que nos encontramos sobre el mapa, y la posición que van siguiendo los enemigos, así como el resto de elementos que hemos colocado. Para volver a controlar nuestro personaje, basta con pulsar de nuevo el icono. Este objeto está formado por una camára con perspectiva 3D situado arriba y enfoncando el mapa entero. Como hijo del objeto "**Camara general**", se encuentra "**FlechaProtagonista**", el está formado por un GUITexture con forma de flecha blanca y un scrpit llamado "**ObjectLabel.js**", el cual se encarga de dibujar en pantalla la flecha en la posición en la que se encuentra nuestro personaje.

- **Camara info:** se trata de una cámara cuya función es que determinados elementos se dibujen en pantalla, superponiéndose sobre los elementos que dibuja la cámara principal del juego. En este caso, los elementos que dibujará esta cámara será el texto que se muestra encima del papel con el clip y que indican la información de la partida: nivel, puntuación y enemigos restantes.
  Este efecto permite de varias cámaras puedan recoger diferentes elementos y fusionarlos en la imagen final que se mostrará al usuario. La propiedad Culling Mask de la cámara será la que indique que deberá representar y qué no. Para decírselo, simplemente seleccionamos los layers que queremos que tenga en cuenta el Culling Mask de la cámara en el editor.

- **Camera Relative Controls:** Unity incluye varios paquetes de controladores standard ya programados. Existen varios, dependiendo del tipo de juego que deseamos crear: si es un juego en primera persona, en tercera persona… etc. En nuestro caso, accederemos a una serie de paquetes propios para ser manejados a través de una pantalla táctil. Estos son los **Standard Assets (Mobile)**. Al cargar este paquete, nos encontramos que en la carpeta "**Prefabs**" ubicamos un controlador llamado "**Camera Relative Controls**". Este controlador se basa en un videojuego tipo tercera persona manejado por 2 Joysticks: el de la izquierda servirá para mover a nuestro personaje y el de la derecha para mover la cámara entorno a éste.

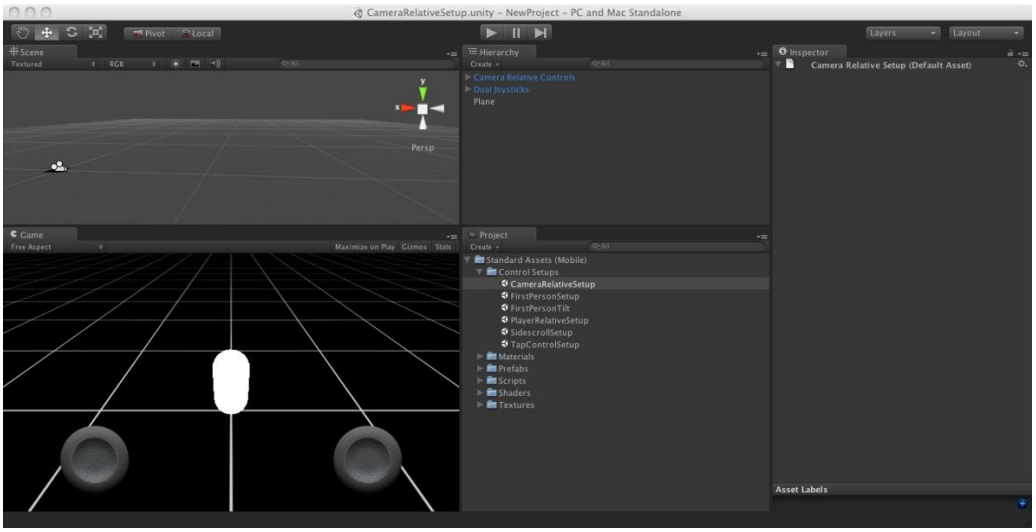


Figura 48: Implementación del paquete Standard Assets (Mobile) [34]

Arriba en la jerarquía podemos observar cómo se ha implementado dicho paquete y como es el resultado en la pantalla de juego

En el caso de nuestro proyecto, vamos a especificar algunas pequeñas modificaciones de dicho paquete que se han realizado.

Como nuestra intención era poder controlar al personaje con una vista más general y desde arriba, hemos modificado la posición de la cámara estando lo más alejada posible del personaje y situada mucho más arriba. Además, se ha suprimido el Joystick que cumplía la función de mover la cámara de un lado a otro, ya que en Contain the Monsters, el personaje se moverá por el mapa mientras la cámara le va siguiendo automáticamente en la dirección en la que éste se va moviendo.

Figura 49: Jerarquía de "Camera Relative Controls"

Los elementos más importantes en la jerarquía de este controlador son, lo que hemos llamado la "**Cámara principal del juego**", el cual hemos hecho ciertas modificaciones como configurar una vista de proyección ortográfica, de forma que le da un aspecto más 2D al juego.

En "**Player**", le asignamos como hijo el objeto "**Protagonista**" que será el personaje de nuestro videojuego, el cual ya vendrá compuesto de las diferentes partes que integran el modelo 3D, así como una lista de animaciones que vienen especificadas cuando nos lo descargamos en el Asset Store.

Figura 50: Jerarquía del "Player"

Destacar que dentro del objeto padre "**Protagonista**", además de las partes que ya incorpora, nosotros le integraremos un objeto llamado "**ConstructorTorreta**", el cual contendrá un script llamado "**crearTorreta.js**" y que permitirá que al pulsar el botón rojo situado a la izquierda, se instancie un clon del modelo "**torreta**" qué posteriormente hemos creado en la posición exacta donde se encuentra nuestro personaje.

[34] *Imagen obtenida desde* http://forum.unity3d.com/ *en Junio de 2014*

Siguiendo con algunas de las modificaciones realizadas en el controlador "**Camera Relative Controls**", hemos desactivado el Joystick derecho de la cámara, como hemos comentado anteriormente, y nos hemos traído el Joystick izquierdo en la zona derecha de la pantalla, de tal forma que está pensado para que el usuario pueda mover al personaje cogiendo el móvil con la mano derecha y usando su dedo pulgar.

- **Controlador del juego:** como su nombre indica, es el objeto encargado de controlar los diferentes estados por los que va atravesando el juego.
  Está formado por 2 scripts: uno llamado "**EstadoJuego.js**" encargado de controlar la información cada nivel, como es el nivel en el que nos encontramos, la puntuación y los enemigos restantes. Además controla el momento en el que el número de enemigos restantes llega a 0, y es entonces cuando tiene que generar una nueva oleada.

  El otro script llamado "**GameOver.js**" es el encargado de activar la **Camara gameover** cuando el juego ha llegado a su fin, así como de mostrar los mensajes de **GAME OVER** con la puntuación alcanzada. Además, se encarga de actualizar y almacenar los valores de puntuación máxima y oleada máxima alcanzada en dicha partida, y que se mostrarán al principio en la pantalla de inicio.

  Por otro lado, tendrá un objeto hijo llamado "**Generador de enemigos**". Dicho objeto a su vez contendrá un script llamado "**GeneradorDeMonstruos.js**", que tiene como misión instanciar los diferentes monstruos que irán apareciendo en las distintas hordas o niveles.

- **GUIs:** se las conoce como **Graphical User Interface** o en español, interfaz gráfica de usuario y se trata de un objeto que reunirá todos los elementos y objetos gráficos para representar la información y acciones disponibles en la interfaz del juego.

Figura 51: Jerarquía de las "GUIs"

En primer lugar, vamos a empezar por los botones situados abajo a la izquierda.

- ✓ "**BotónTorreta**", se tratará del botón rojo en cuyo interior se dibuja la silueta de una torreta. A este objeto se le asocia el script "touchTorreta.js" y tiene como finalidad comprobar si tenemos torretas recogidas disponibles para poder colocar. En tal caso, si pulsamos el botón, colocará la torreta, decrementará el número total de torretas disponibles y actualizará su valor.
  El dibujo de la torreta que se encuentra en el botón se ha obtenido de la siguiente imagen: http://i56.tinypic.com/2niu1ap.jpg

- ✓ El objeto hijo "**NumeroTorretas**" será el GUIText encargado de visualizar debajo del botón el número de torretas disponibles y llevará asociado el script "**contadorTorreta.js**". Este script le servirá al script "**touchTorreta.js**" y a otros scripts para llamar a sus métodos, cuya función es controlar el número de torretas disponibles, según cada caso.

- ✓ El objeto "**BotonValla**" es prácticamente igual al anterior. Está asociado al botón amarillo que sirve para colocar la valla, compuesto de un script "**touchValla.js**" cuya función es igual a la de "**touchTorreta.js**" y un objeto hijo llamado "**NumeroVallas**" también idéntico al anterior, con un script asociado llamado "**contadorValla.js**"

- ✓ "**GUI vidaCasas**" hace referencia al icono de la casa con la barra de vida que se encuentra situado arriba a la derecha. Será el objeto encargado de dibujar la barra de vida de nuestra aldea, es decir, su nivel de destrucción, así como también servir de botón para que al pulsar se active la **cámara general**, en cuyo caso podremos tener una vista general de todo el mapa. Por tanto, a este objeto se le asociarán los scripts "**VidaCasas.js**" para dibujar en cada momento el estado de la aldea, y el script "**touchCamaraGeneral.js**", para cambiar de una cámara a otra.

  El dibujo de la casa ha sido obtenido de:
  http://rlv.zcache.es/casa_del_dibujo_animado_comunicado_personalizado-r351f6bf09fe045feb4dd8aa56fe52c5c_imtq3_8byvr_512.jpg

- ✓ "**GUI vidaPlayer**" es el objeto encargado de representar las vidas del jugador. Al iniciar la partida tendremos 3 vidas y a medida que recojamos corazones, podremos ir sumando vidas hasta un máximo de 5. Si los enemigos nos tocan o atacan, se restarán las vidas hasta llegar al fin del juego. Lleva asociado el script "**vidaPlayer**", que es el encargado de que incrementar, restar, resetear o actualizar las vidas, así como su representación. Cada uno de sus hijos serán los iconos de vida que se dibujan en pantalla, tipo GUITexture.

- ✓ Seguidamente describiremos el objeto "**InfoJuego**". Se trata del objeto encargado de mostrar la información real acerca del estado de la partida. Esta información podemos encontrarla arriba a la izquierda, escrita encima de un fondo con forma de hoja de papel agarrada con un clip, obtenida de Imágenes

Google. Dicha hoja será el objeto "**FondoGui**" y el objeto "**InfoGui**" será padre de los objetos encargados de mostrar el texto de esta información. Cabe destacar que ninguno de los objetos que conforman "**InfoJuego**" contiene ningún script, ya que su modificación será realizada por medio de otros Scripts que tendrán acceso desde otros puntos distintos.

✓ Por último describimos el objeto "**Menú principal**", el cual se trata de un GUITexture con forma de cartel, situado en la parte superior derecha de "**InfoJuego**" y que nos permite que, al seleccionarlo con el dedo, se abra un pequeño menú donde podamos elegir entre continuar la partida, reiniciarla y comenzar desde el principio o salirse del juego.

El dibujo fue obtenido de: http://www.colorearjunior.com/dibujos-para-pintar-de-letrero-de-madera-en-la-monta%C3%B1a-se%C3%B1al_10114.html con sus respectivas modificaciones.

"**Menú principal**" está compuesto de un hijo "**menú**" que a su vez contiene 3 hijos de tipo GUIText que son las 3 opciones que se mostrarán en pantalla cuando pulsemos el cartel de menú. En el momento en que aparecen estas opciones, queda pausado el juego.

Estos objetos serán:

- Continuar: lleva asociado el script "**touchContinuar.js**" y contiene las instrucciones para que al pulsar el GUIText se reanude la partida.

- Reiniciar: lleva asociado el script "**touchReiniciar.js**" y contiene las instrucciones necesarias que al pulsar el GUIText se resetee la partida y empiece de nuevo en la primera oleada. Desde este script se invocan también los scripts "**EstadoJuego.js**" y "**Reseteador.js**", encargados de invocar las funciones necesarias para que se reinicien todos los valores y objetos que participan en la partida

- Salir: lleva asociado el script "**touchSalir.js**" y contiene las instrucciones necesarias para que al pulsar el GUIText se salga del juego.

- Fondo: se trata de la imagen de fondo que se visualiza detrás de las opciones y que imita a un pos-it pegado con un celo. La imagen ha sido obtenido a través de la siguiente URL: http://s281.photobucket.com/user/kidjay90/media/Sticky-Note.png.html

Figura 52: Opciones que se despliegan tras pulsar el icono "Menú"

- **Luz direccional:** es la luz encargada de iluminar todos los elementos 3D que intervienen en la escena del juego.

- **Fábrica de preguntas:** Reúne la parte educativa. Se trata del objeto encargado de la lectura del fichero "**plantilla.xml**", contenedor del listado de preguntas que se lanzarán al jugador cuando este pase por encima de un cubo de color, así como de representar dichas preguntas en forma de una interfaz amigable y que permita al jugador seleccionar una de las tres respuestas. Posteriormente, se mostrará un mensaje en pantalla que indique al jugador si la respuesta elegida es la correcta o no. Dicho objeto permanecerá desactivado hasta que el jugador pase por encima de un cubo. Una vez que el cubo es tocado por el personaje, el objeto "**Fabrica de preguntas**" se activará representando la pregunta en pantalla y se desactivará cuando el jugador haya terminado de responder.

  El objeto "**Fabrica de preguntas**" se compone de 2 scripts:

  - ✓ **LectorXML.js**: se encarga de importar las librerías necesarias para poder leer el fichero "**plantilla.xml**" y obtener la información contenida en los nodos. Su principal función es acceder a la ruta donde se encuentra ubicado el fichero XML y ser capaz de interpretar su contenido para posteriormente pasarle la información al script **preguntaGUI.js**.

Su funcionamiento es el siguiente: al ejecutarse, se elige un nodo correspondiente a una pregunta aleatoria y se almacena en una serie de variables la información contenida de dicho nodo: pregunta elegida, tres respuestas posibles y respuesta correcta. Además, almacena a través de dos variables booleanas si el cubo sobre el que se ha pasado es rojo o amarillo. El valor de estas variables será recogido por el script "**preguntaGUI.js**"

- ✓ **PreguntaGUI.js**: este script tiene la función de representar gráficamente la interfaz que lanzará la pregunta, así como de mostrar si ha sido correcta o no la respuesta y aplicar el resultado según corresponda. Para ello, lo que hace en primer lugar es obtener el valor de las variables correspondientes a la pregunta elegida, almacenadas en el script anterior, así como las variables que almacenan el número de vallas o torretas que disponemos en nuestro inventario. Posteriormente se creará el interfaz con la función **OnGUI()**, mostrando el enunciado con las 3 posibles respuestas. Estas respuestas se mostrarán sobre un fondo con forma de papeles que le dará al interfaz un toque más amigable de cara al usuario. Tanto los enunciados, como las respuestas y las ventanas de resultados tendrán una apariencia que se configurará a través de una serie de parámetros almacenados en variables de tipo **GUIStyle**. Posteriormente comprobará si la respuesta elegida es la correcta o no y según sea, se lanzará una ventana con el resultado. Si la respuesta ha sido la correcta, se comprobará cual ha sido el cubo recogido. Dependiendo del color del cubo, accederemos e incrementaremos la variable que corresponde al número de vallas o al número de torretas según si el cubo es amarillo o rojo. De esta forma, dispondremos de un elemento defensivo más (ítem). En cambio, si la respuesta elegida no es la correcta, no se incrementará ninguna de las variables. De esta forma, seguiremos teniendo el mismo número de torretas y vallas.

En ambos casos, una vez formulada y contestada la pregunta, el cubo quedará desactivado para que el jugador no lo vea y no pueda recogerlo más veces.

Las texturas e imágenes que aparecen en la interfaz, han sido obtenidas de Google Imágenes.

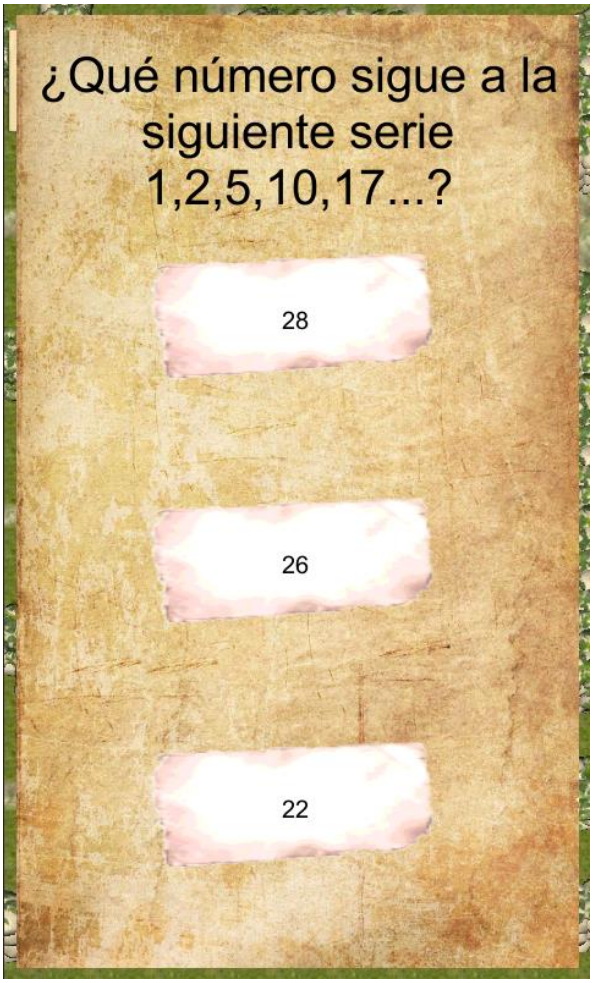


Figura 53: Interfaz de preguntas

Figura 54: Ventana de resultado

4.1.1.3 Objetos comunes

Se trata de aquellos elementos que intervienen en la pantalla de juego y la pantalla principal. Son elementos que se cargan una única vez y siempre se encontrarán activados, tanto en el inicio como en el transcurso de la partida. Esto permitirá un ahorro considerable de la memoria y evitará duplicados innecesarios.
Vamos a ir viendo con detalle cada uno de los elementos:

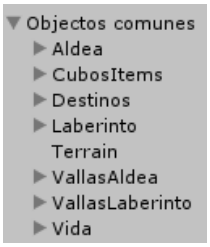


Figura 55: Jerarquía de "Objetos comunes"

- **Aldea:** lo conforman diferentes modelos de casas, así como una torre de guardia situada en el centro. Los modelos han sido descargados del Asset Store:

    - Torre de guardia:
    Low PollyRTD Orc Tower
    Categoria: 3D Models> Enviroments
    Publicador: BITGEM
    - Casas:
    Customizable Medieval Houses
    Categoria: 3D Models> Enviroments
    Publicador: Xocolatl Studio

  A todos ellos se les ha incorporado un Box Collider para detectar el momento en que los monstruos llegan a ellas y "colisionan".

- **CubosItems**: contiene todos los objetos "**cubo**" repartidos por el mapa, tanto los amarillos como los rojos. Ambos tipos de cubo llevan asociados varios scripts. En común tienen un script llamado "**RotarObjeto.js**" que se utiliza para que el objeto gire permanentemente sobre sí mismo. Además incorporan un Box Colider para detectar cuando el personaje pasa sobre ellos. Los cubos amarillos contienen el script "**CogerItemValla.js**", así como los rojos contienen el script "**CogerItemTorreta.js**" y su función es lanzar las preguntas en el momento en que el jugador pasa por los cubos. En el momento en que la pregunta ha sido formulada, el cubo desaparece.

- **Destinos**: contienen una serie de objetos que hemos llamado "**Pivotes**" enumerados del 00 al 44 donde la primera cifra empezando por la izquierda hace referencia a las columnas y la segunda a las filas, así como otros 5 pivotes llamados c1, c2, c3, c4 y c5. Para entender cuál es la función exacta de dichos pivotes, conviene leer el apartado **4.2.3.1 IA de los enemigos: funcionamiento de los Nodos**. Se trata de objetos con forma de esfera que no se renderizan, es decir, que el usuario al jugar no tiene constancia de ellos porque no los ve, son invisibles. En cambio tienen una función fundamental y es guiar a los monstruos para que sepan a qué puntos del mapa deben dirigirse. Son los puntos destino por los que los monstruos deberán transitar, pasando de unos a otros, hasta llegar a los pivotes finales del mapa: c1, c2, c3, c4 y c5, qué son los correspondientes a cada una de las casas que forman nuestra aldea.

- **Laberinto:** conforma el conjunto de objetos con forma de muro de piedra y vegetación que dibujan el laberinto. Tienen como finalidad el separar los diferentes caminos por los que van a transitar los monstruos.

El modelo 3D ha sido obtenido del Asset Store:
Dry Stone Wall with Leafy Vines
Categoria: 3D Models>Props>Exterior
Publicador: bitsong

- **Terrain**: se trata del terreno o suelo sobre el que se van a situar todos los elementos del escenario. Unity incorpora un editor muy completo de terrenos. Se ha tratado el terreno creando una serie de montañas alrededor del laberinto que sirven para adornar y delimitar nuestro escenario. Posteriormente se ha aplicado una texturización del mismo, proporcionándole al suelo un tono verdoso y a las montañas una textura rocosa de color marrón y blanco para darle un pequeño efecto nevado.

- **VallasAldea**: Son las vallas que rodean nuestra aldea y la protegen temporalmente. Son un modelo distinto al modelo que colocará nuestro personaje por el laberinto para retener a los monstruos. En cambio cumplen la misma función y aparecerán por defecto cuando iniciemos una partida. Cada valla contiene un Box Collider para detectar cuando un monstruo llega y se acerca a ellas, atacándolas.

  El modelo 3D ha sido obtenido gratuitamente del Asset Store:
  Stone Fence
  Categoria: 3D Models>Props>Exterior
  Publicador: Ben Droste

- **VallasLaberinto**: Es el objeto que contiene todas las vallas que nuestro personaje podrá ir colocando a lo largo del laberinto para retener a los monstruos y evitar que lleguen a nuestra aldea. Para entender su funcionamiento, vamos a analizar cada uno de los objetos hijo que contiene:

Figura 56: Jerarquía de "VallasLaberinto"

Cada objeto valla contiene como padre un objeto llamado "**vallaTrigger"**. Los Triggers no son otra cosa que disparadores o activadores de eventos. La función que cumple **vallaTrigger** es que cuando nuestro personaje pise una zona cercana a las líneas que delimitan los lugares donde podemos colocar una valla, se active el botón amarillo correspondiente a la posibilidad de colocar una valla.

Es entonces cuando el usuario decidirá si colocar o no en ese lugar una valla.

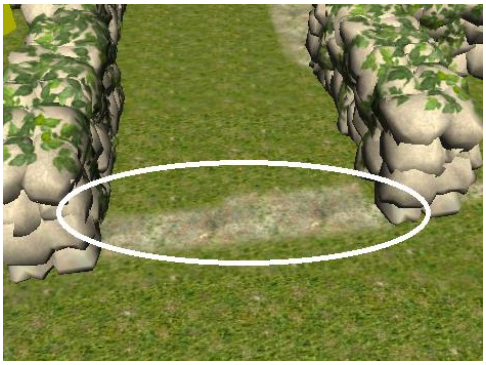


Figura 57: Línea de colocación de vallas

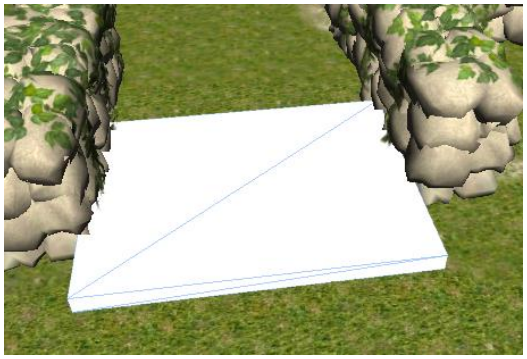


Figura 58: Objeto "vallaTrigger" renderizado

En el dibujo de la izquierda podemos apreciar la zona que delimita donde podremos colocar la valla. En el dibujo de la derecha podemos apreciar el objeto "**vallaTrigger**" renderizado para su comprensión. Recordar que en el juego, este objeto es invisible al usuario, es decir, no se renderiza. Cuando nuestro personaje toque este objeto, se disparará el evento de ofrecer al usuario la posibilidad de colocar allí una valla, iluminándose el botón amarillo.
Si el usuario dispone de vallas en su inventario, pulsará el botón y el objeto valla, que anteriormente se encontraba desactivado como podemos ver en la jerarquía, se activará.

El objeto **vallaTrigger** lleva asociado un script llamado "**ActivadorObtaculo.js**" que será el encargado de todo lo comentado anteriormente. Una vez que el objeto "**valla**" se active, a su vez se activará el script asociado "**VidaObtaculo.js**" que será el encargado de controlar la vida de la valla o resistencia ante el ataque de los monstruos. Esto quiere decir que la valla tendrá una vida con valor 100 y a medida que sea atacada por los enemigos, este valor irá disminuyendo hasta llegar a cero, en cuyo caso la valla será destruida y se desactivará. Al destruirse, **se activará un sistema de partículas de polvo** eventual, producto de la destrucción. Más tarde, si nuestro personaje dispone de vallas, podrá colocarlas de nuevo en el mismo lugar o en otro lugar distinto dentro de las zonas delimitadas.

- **Vida**: Es el objeto padre que contiene todos los objetos "corazón".

Figura 59: Jerarquía de "Vida"

El objeto corazón representa las vidas que puede recoger el personaje a lo largo del escenario para poder sobrevivir. Son representados con forma de corazón y cada uno de ellos lleva asociado 2 scripts:

"**RotarObjeto.js**" que se encarga de hacer rotar de forma repetitiva el corazón.
"**CogerVida.js**" que se encarga de sumar una vida al personaje cuando este pasa por ellas. Se limita además a comprobar el número de vidas que tiene el personaje. En caso de que tener 5 vidas, no podremos recoger más.

El modelo 3D del corazón ha sido obtenido de forma gratuita a través de la siguiente dirección:
http://www.unitymagic.com/shop/en/health-pickup-rotating-heart-full-version.html

4.1.2 Objetos prefabs presentes en el proyecto

Los objetos prefabs u "**Objetos Prefabricados**" son un tipo de GameObject especial con detalles específicos que nos permiten definir las propiedades de un objeto y además poder instanciar una o más veces el mismo.
Una vez que se crea un prefab, duplicarlo es sencillísimo. Los prefabs ahorraran un montón de tiempo ya que es una manera rápida de guardar las configuraciones de cada objeto. A continuación se muestran los diferentes prefabs diseñados en el proyecto, su organización y su funcionamiento. Los dividimos en grupos.

4.1.2.1 Enemigos

Son los diferentes modelos de monstruos que irán apareciendo a lo largo del juego. Estos modelos llevarán asociados los scripts de inteligencia artificial (IA) que se han programado para su comportamiento y que posteriormente explicaremos.
Se almacenan en forma de prefabs y se les instanciará tantas veces como numero de monstruos aparezcan en un nivel. Conforme el nivel de dificultad vaya aumentando, podrán instanciarse varios tipos de monstruos en un mismo nivel teniendo por ejemplo 3 monstruos de un tipo y 3 de otro.

A continuación pasamos a ver los modelos de monstruo empleados, los datos del Asset Store de donde se han obtenido gratuitamente y sus características:

- **Araña 1**

Free Fantasy Spider
Categoria: 3D Models>Characters>Creatures
Publicador: Kalamona
Velocidad: normal
Nivel de salud: 50

Figura 60: Modelo 3D Araña 1

- **Araña 2**

Spider Green
Categoría: 3D Models > Characters > Animals > Insects
Publicador: PollyNext
Velocidad: lenta
Nivel de salud: 80

Figura 61: Modelo 3D Araña 2

- **Monstruo Verde**

Monster2
Categoria: 3D Models > Characters
Publicador: BUMSTRUM
Velocidad: normal
Nivel de salud: 100

Figura 62: Modelo 3D Monstruo Verde

- **Duende**

Goblin
Categoría : 3D Models > Characters > Humanoids > Fantasy
Publicador: PolyNext
Velocidad: normal
Nivel de salud: 150

Figura 63: Modelo 3D Duende maligno

- **Monstruo Azul**

Maze element Ice Golem
Categoría: 3D models>Characters>Humanoids>Fantasy
Publicador: Andres Olivella
Velocidad: rápida
Nivel de salud: 200

Figura 64: Modelo 3D Monstruo Azul

Estos modelos 3D vienen acompañados de una serie de animaciones particulares que serán reproducidas según lo indique la inteligencia artificial (IA) de cada uno de los monstruos.

En cuanto a su comportamiento, todos los enemigos se basan en la misma IA.
La única diferencia a destacar será el valor máximo de salud del monstruo, guardado en el script "**NivelSalud.js**", que se encargará de restar la salud cada vez que una bala de la torreta impacte sobre el mismo hasta llegar a 0, en cuyo caso el monstruo desaparecerá seguido de un sistema de partículas que se lanzará para indicar la eliminación del mismo, llamado "**impacto final**".

En la imagen superior hemos descrito el nombre oficial del modelo 3D, la categoría que ocupa dentro del Asset Store y su publicador. A continuación, también se ha

añadido la velocidad y nivel de salud máxima que se le ha dotado a cada enemigo dentro del juego.

Otro script asociado a cada monstruo será el que permite controlar las animaciones, según el caso. Por ejemplo, cuando uno de los monstruos desea atacar, la IA llamará a este script para indicarle que quiere reproducir la animación de golpear. Como cada modelo de monstruo contiene sus propias animaciones, tendremos 5 scripts distintos que corresponderán a cada monstruo. Estos serán: "**ContrlAnimArania1.js**" para las animaciones de la primera araña, "**CntrlAnimsArania2.js**" para las animaciones de la segunda araña, "**ContrlAnimMonstruoVerde.js**" para las animaciones del monstruo verde, "**ContrlAnimsDuende.js**" para las animaciones del duende maligno y "**ControlAnimMonstruoAzul.js**" para las animaciones del monstruo azul.
A continuación vamos a explicar y desarrollar el funcionamiento de la IA de los monstruos a partir de diagramas de estado. Estos diagramas corresponderán a los estados por los que van pasando los monstruos.

En primer lugar, se necesita incorporar a la IA un código de **pathfinding**, esto es, una forma en la que una entidad en movimiento, como son en nuestro caso los enemigos, encuentra el camino más corto entre dos puntos, también llamados nodos. Se trata por tanto de un algoritmo que permita la búsqueda de rutas en tiempo real para poder alcanzar un destino, eligiendo el camino más óptimo teniendo en cuenta los obstáculos.

Existen diferentes algoritmos de búsqueda populares como por ejemplo el A*, WaveFront, BrushFire… etc, pero Unity ya incluye uno propio muy fácil de usar llamado Navigation.

Unity crea una malla (mesh en inglés) para poder separar los obstáculos del terreno con el fin de que lo enemigos esquiven los obstáculos cuando están en busca del jugador.

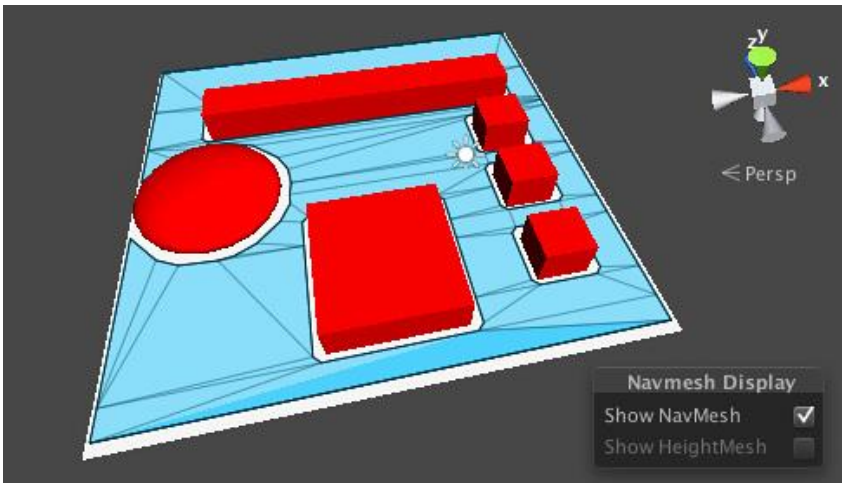


Figura 65: Ejemplo de la malla creada por Unity para detectar obstáculos [35]

[35] *Imagen obtenida desde* https://docs.unity3d.com/Documentation/Images/manual/Navmeshes-2.jpg *en Junio de 2014*

Una vez que **navmesh** está configurado, los enemigos tienen que estar equipados para poder usarlo. A estos se les incorporará el componente de navegación llamado **Nav Mesh Agent**. Este componente gestiona tanto la búsqueda de caminos como el control de los movimientos de un personaje. Por lo tanto, lo primero que se hará en el script de la IA de los monstruos será agregar este componente. Más adelante se configuraran sus propiedades en el inspector.

Una vez que me ha configurado el sistema de navegación que usará Unity, será necesario indicar los nodos destino hacia los que se dirigirán los monstruos para atravesar el camino. Estos tienen como objetivo alcanzar la aldea como bien hemos explicado, pero para dar más aleatoriedad al juego se ha decidido colocar diferentes nodos llamados **pivotes**, hacia los que se irán dirigiendo los enemigos. Estos nodos servirán de puntos intermedios por los cuales los monstruos irán pasando de uno a otro, siendo elegidos de forma totalmente aleatoria hasta alcanzar el nodo final que se situará en la aldea.

Para entender su funcionamiento, vamos a exponer la siguiente captura:

Figura 66: Mapa del juego con los nodos "pivote" enumerados

En esta imagen se puede apreciar los nodos "pivote" con forma de esfera blanca, los cuales han sido renderizados para una más fácil comprensión. Como puede apreciarse, si vemos el mapa desde arriba, los nodos están dispuestos espacialmente en grupos de 6 filas, 5 situadas en el laberinto y una situada en la aldea. Cada fila tiene a su vez un grupo de 5 pivotes que según la posición que ocupan podemos dividirlos también en 5 columnas. Estas divisiones espaciales pueden verse reflejadas en la numeración de cada nodo, siendo las unidades correspondientes a las columnas y las decenas a las filas.

Aquí se puede apreciar con más detalle las divisiones espaciales de cada nodo.

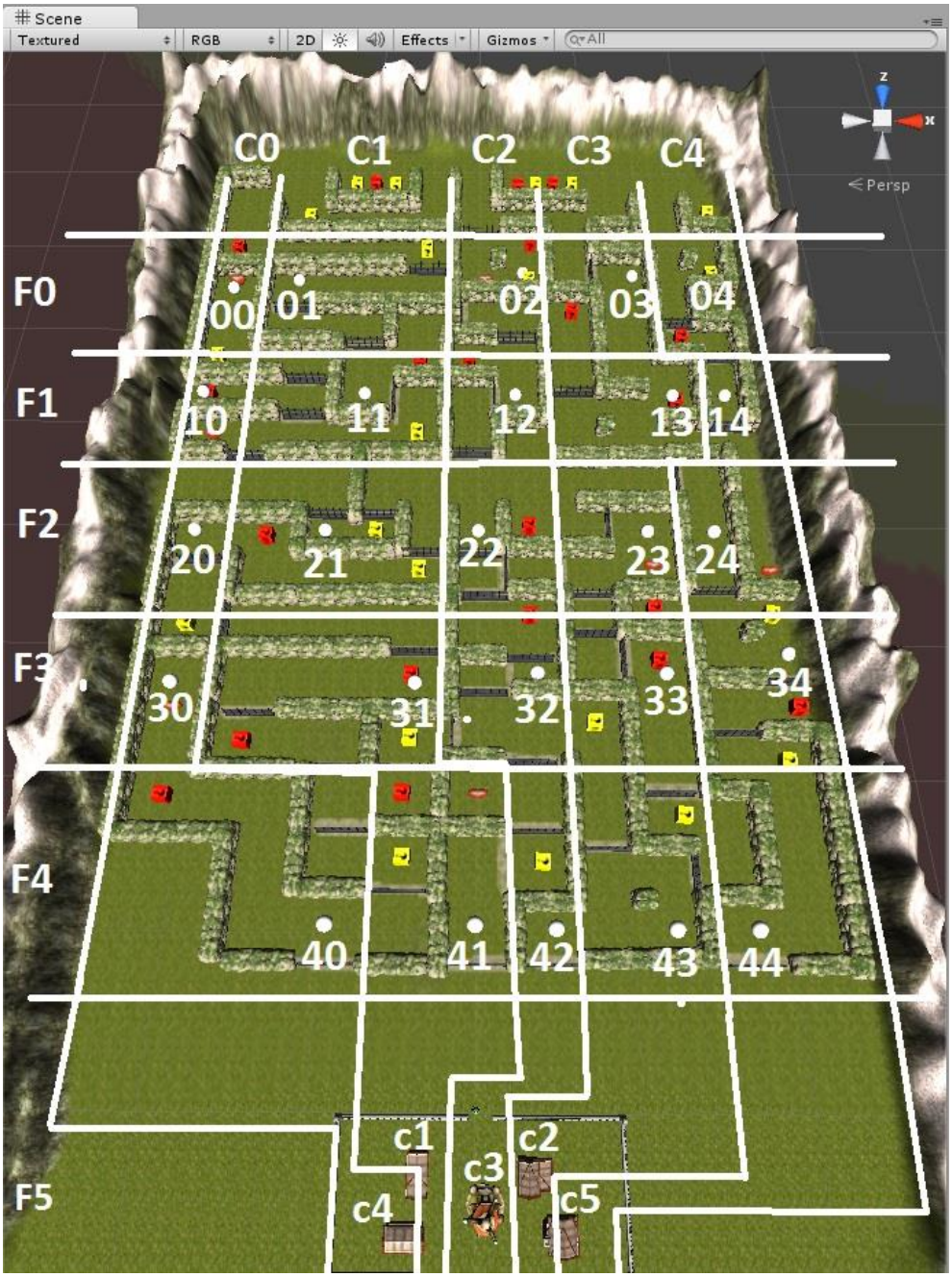


Figura 67: Mapa del juego dividido en filas y columnas

Por tanto los monstruos, una vez que aparecen, irán recorriendo los nodos dirigiéndose primero a la fila 1 y eligiendo de forma aleatoria hacia qué nodo de la fila 1 se moverán. Una vez alcanzado este, se desplazarán a la siguiente fila de nodos, eligiendo de nuevo aleatoriamente otro nodo de la nueva fila. Este proceso se repetirá de forma sucesiva, avanzando por el laberinto hasta alcanzar los nodos c1, c2, c3, c4 o c5, correspondientes a la aldea. Una que se hayan alcanzado dichos nodos, los monstruos habrán completado su desplazamiento y comenzarán a atacarla.

A lo largo de este desplazamiento irán pasando por diferentes estados que vamos a mostrar en forma de diagrama de estados para una mejor comprensión.

A continuación se va a mostrar los 4 estados por los que atraviesa la IA del enemigo.

Estado: Patrulla

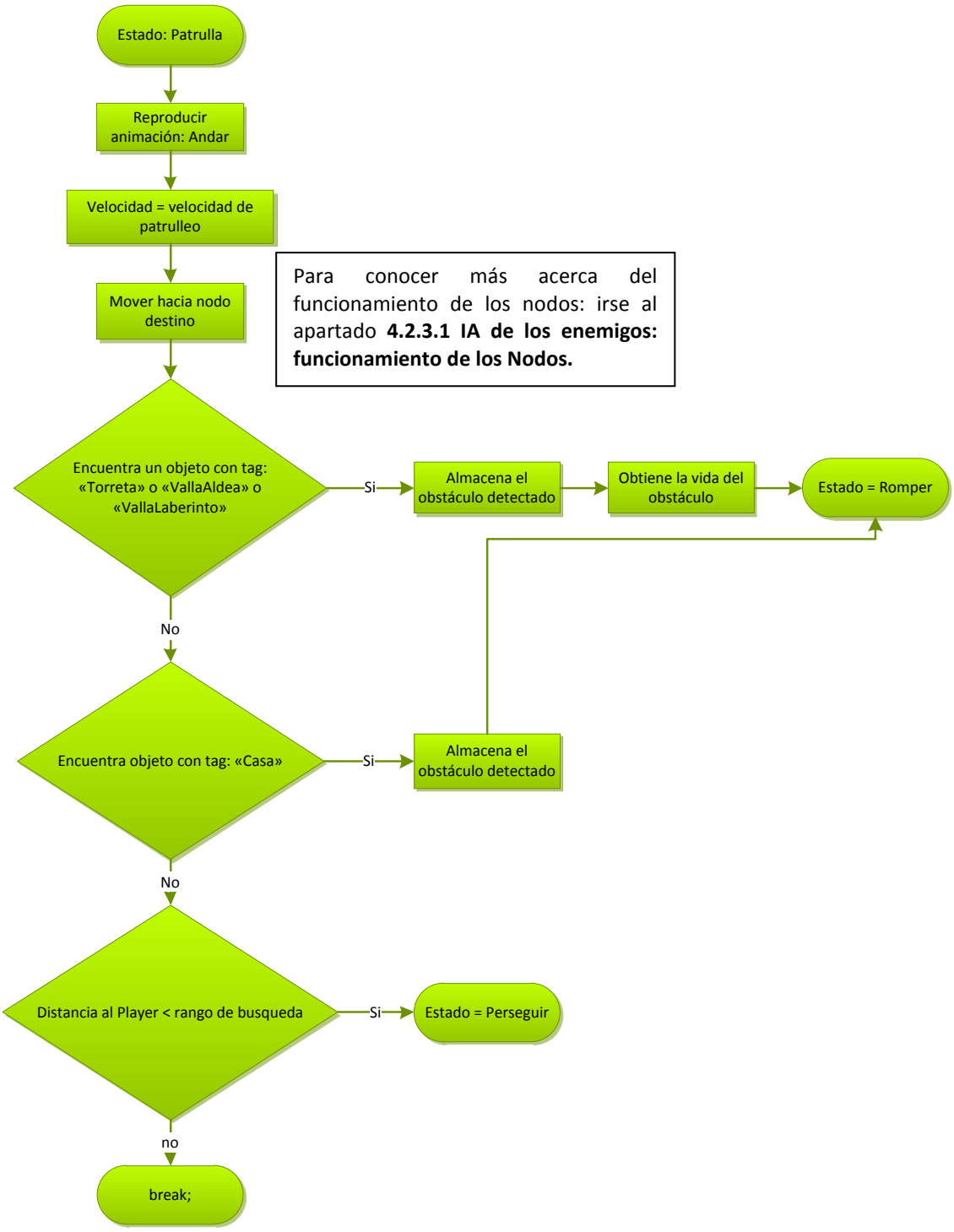


Figura 68: Diagramas de estados: estado "Patrulla"

Estado: Perseguir

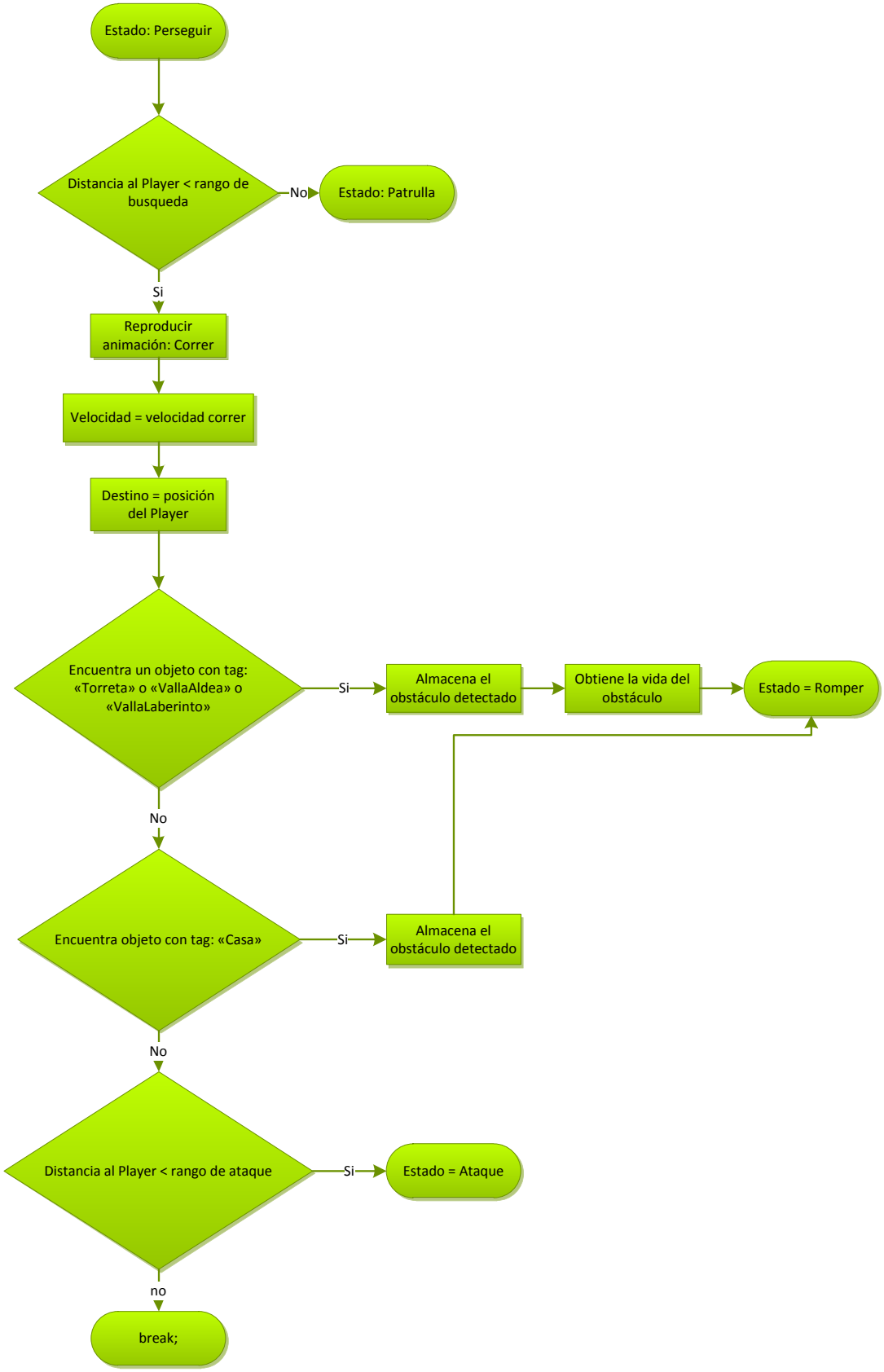


Figura 69: Diagramas de estados: estado "Perseguir"

Estado: Ataque

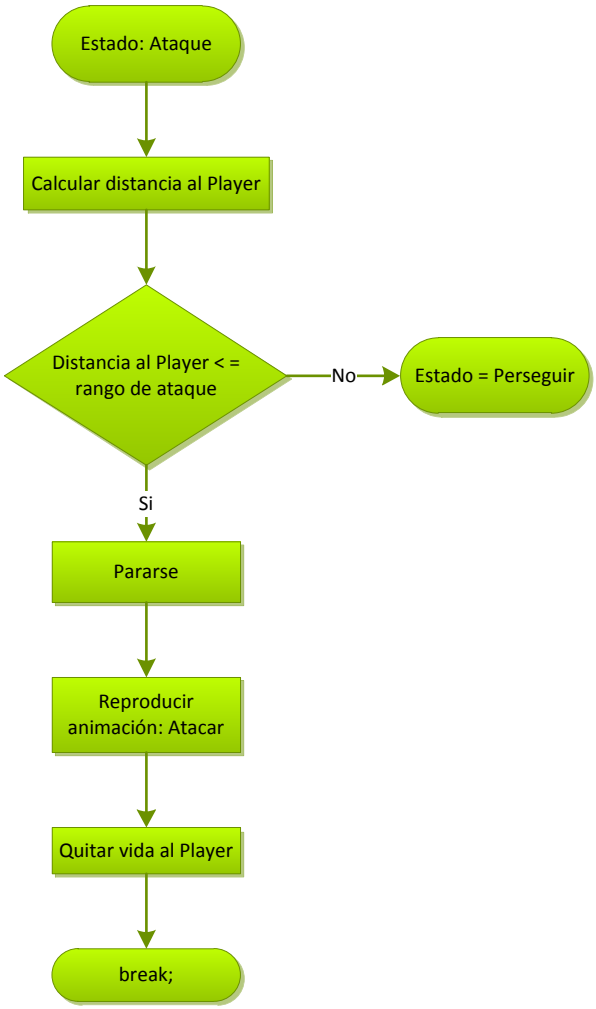


Figura 70: Diagramas de estados: Estado "Ataque"

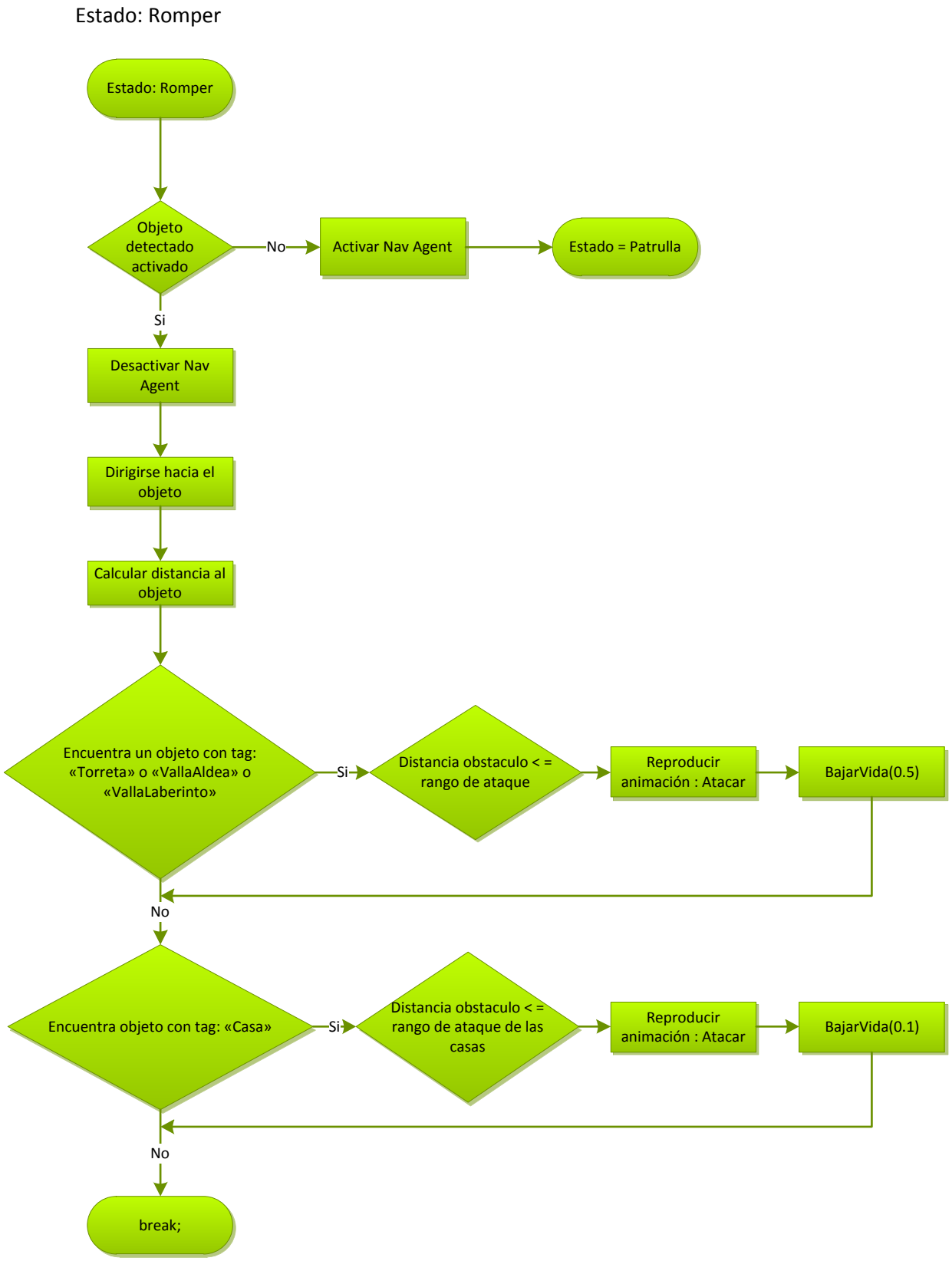


Figura 71: Diagramas de estados: Estado Romper

4.1.2.2 Torretas

El objetos prefab principal que se utilizará en el juego son las torretas que el personaje principal colocará por el mapa. Estas se instanciarán cada vez que el jugador disponga de torretas en su inventario y desee pulsar el botón de color amarillo.
Se utilizará en el juego como arma automática para poder acabar con los monstruos que van atravesando el laberinto.

A este modelo 3D, obtenido del Asset Store:

Nombre: WCE - Turret

Categoría: 3D Models > Props >Weapons

Publicador: Piflik

Figura 72: Modelo 3D torreta

Pueden instanciarse tantas como número de ellas tengamos en nuestro inventario. Su comportamiento se basa principalmente en vigilar en un radio de 70 metros. La torreta solo podrá girar sobre sí misma. En el momento en que los monstruos entren dentro de esta área, la torreta les apuntará y comenzará a disparar sobre ellos. Si varios monstruos entran en este radio, la torreta escogerá a uno, normalmente al primero que detecta y empezará a disparar hasta acabar con su vida. La torreta dejará de atacar cuando ésta consiga acabar con la vida del monstruo o hasta que éste se aleje del radio de ataque.

A continuación vamos a analizar cada una de las partes y objetos de las que está compuesta la torreta. Al modelo 3D se le han añadido los siguientes objetos hijos:

- **BaseTorreta**: debido a que la torreta girará sobre si misma se ha decidido crear una base compuesta de un cilindro simple de color blanco que representa la base sobre la que girará.

- **FocoLuz**: una luz direccional de tipo Spot que representará un foco de luz donde la torreta apunta para vigilar y disparar.

- **EsferaDetectora**: Se trata de una esfera invisible que no se renderiza y que lleva asociada un script llamado "**esferaDetectora.js**" cuya misión es detectar los enemigos que entran dentro de la misma. Es el método que se ha desarrollado para que la torreta pueda detectar la presencia de monstruos alrededor de un

perímetro. De esta forma, la esfera representa el radio de acción sobre el cual la torreta podrá actuar para disparar. En el momento en que los monstruos la atraviesan, el script "**esferaDetectora.js**" se encargará de pasarle a la IA de la torreta, el target u objetivo sobre el cual atacar.

- **Lanzador**: Es el objeto encargado de instanciar los objetos **bala**. Tiene como misión ejecutar el disparo hacia el enemigo. Contiene el script "**Disparo.js**" encargado de instanciar además de las balas, el sistema de partículas "**destello**" que será la explosión que genera el cañón de la torreta al disparar, así como la configuración de la velocidad de la bala.

Debajo se muestra en la imagen de los elementos que componen la torreta:

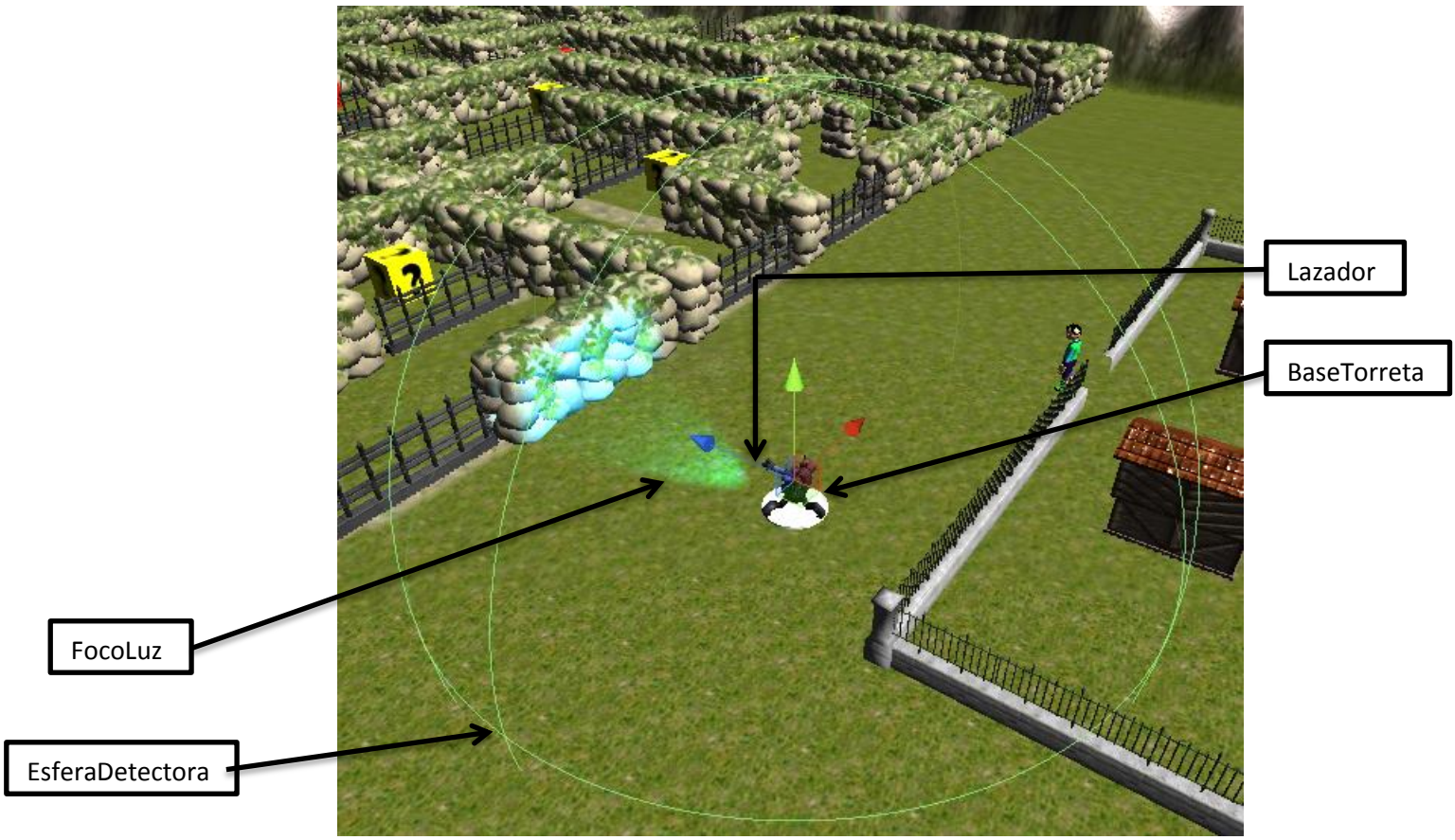


Figura 73: Elementos que componen el objeto torreta

A continuación se va a mostrar los diferentes estados y comportamientos de la IA (inteligencia artificial) de la torreta contenidos en el Update.

Antes de indicar estos estados, vamos a mostrar la función Start() para comprobar las variables que se inicializarán nada más arrancar el script.

```
function Start (){

// Instanciamos el script "Disparo.js" que se ejecutará más tarde en
el estado de ataque y del que se invocará el método Disparar()
disparoScript = gameObject.GetComponentInChildren(Disparo);

// Inicializamos variables de rotación de torreta
rotacion = Random.Range(0,180);// (0-180)
rotActual = 0;

// Comprobamos que tengamos un blanco al que buscar y disparar
if(!target){
        return;
    }

}
```

Estados y comportamientos de la IA (inteligencia artificial) de la torreta contenidos en el método Update():

- **Estado de Vigilancia:**
  En este estado la torreta girará sobre sí misma en posiciones aleatorias, simulando que se encuentra vigilando su alrededor en busca de enemigos.
  Tendrá lugar una rotación sobre sí misma en ángulos aleatorios que irán desde -90° a 90° girando sobre su eje Y. Por cada desplazamiento en la rotación, se hará una pausa de 1 segundo.

  Para su funcionamiento, en este estado se toman las variables "**rotacion**", que será un valor aleatorio en grados de la rotación que deberá hacer la torreta sobre su eje Y, y "**rotActual**" que será la rotación que tiene actualmente la torreta en este momento. El valor del que parte nada más arrancar su IA será **rotActual = 0** y la variable rotación tomará un valor random. En el momento en que un enemigo entre en contacto con la **esfera detectora**, se le asignará ese enemigo a la variable **target** y acto seguido se pasará al estado de ataque.

- **Estado de Ataque:**
  En este estado la torreta comprobará a través del método **VeoAlEnemigo()** la distancia que hay al enemigo y en caso de que sea menor que el rango de ataque, comenzará a disparar, ejecutando cada disparo en un determinado periodo de tiempo.

A continuación mostramos un diagrama de flujo de estos 2 estados:

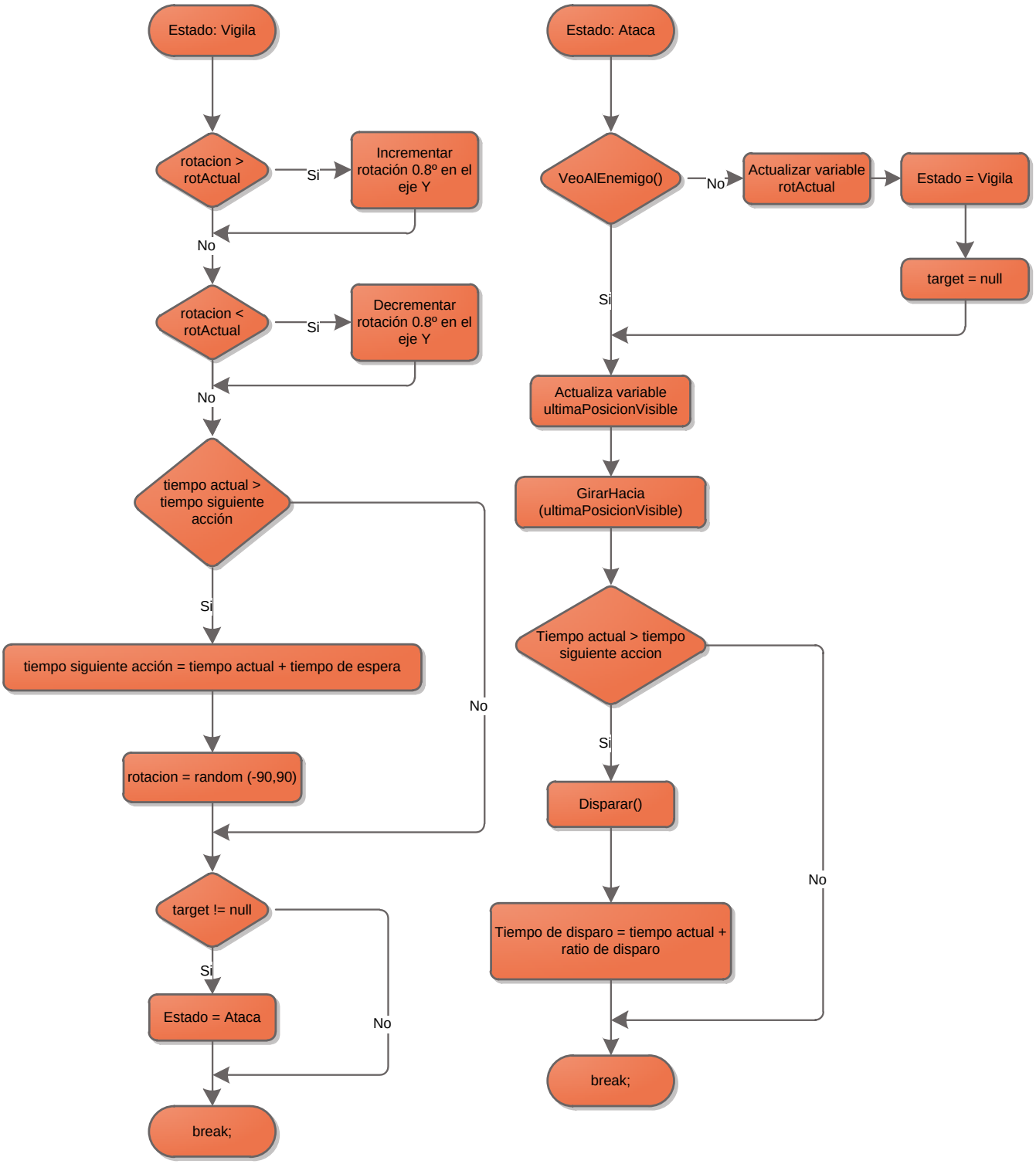


Figura 74: Diagramas de flujo de los estados "Vigila" y "Ataca" de la torreta

- **Bala:** Se trata de un objeto prefab que será instanciado por el Lanzador cada vez que se ejecute un disparo.

  Geométricamente se trata de una capsula simple de color blanca. A este objeto se le añade el componente **Rigidbody**. La clase **Rigidbody** controla la posición de un objeto a través de simulación de física. Este componente, al ser añadido a un game object, toma el control sobre la posición de un objeto, haciendo que caiga bajo la influencia de la gravedad, y puede calcular cómo los objetos responderán a las colisiones. En nuestro caso, daremos a la bala también estas propiedades para poder simular un disparo real. Por otro lado, la bala lleva asociado un script llamado "**DestruirBala.js**" cuya misión será instanciar un sistema de partículas cada vez que se produce un impacto de la bala en los monstruos. Además, este script se encargará de destruir los objetos "bala" cuando el impacto haya tenido lugar.

4.1.2.3 Efectos de partículas

Reúne todos los sistemas de partículas aplicados en cada una de las situaciones. Vamos a describir cada uno de ellos:

- **Destello**: es el sistema de partículas que se ejecuta cuando la torreta dispara.

Figura 75: Representación del sistema de partículas "destello"

Podría considerarse como el fogonazo que produce el cañón de la torreta cuando dispara. Se le asignó este sistema de partículas para dar mayor realismo y un mayor énfasis al comportamiento cuando la torreta dispara. Este sistema de partículas procede de los Standard Assets propios de Unity.

- **Impacto**: se trata del sistema de partículas que se instancia cuando la bala impacta sobre uno de los monstruos.

Figura 76: Representación del sistema de partículas "impacto"

Sirve para que el jugador reconozca con más exactitud cuando estamos haciendo daño a los monstruos.
Este sistema de partículas ha sido obtenido gratuitamente a través del paquete "FT Free Sample" del Asset Store cuyo publicador se llama FlyingTeapot.

- **Impacto final**: es el sistema de partículas que se instanciará cuando la vida del monstruo llegue a cero.

Figura 77: Representación del sistema de partículas "impacto final"

Servirá para que el jugador pueda apreciar con un mayor realismo cuando se ha acabado con un monstruo. En el caso de la imagen, puede apreciarse como la torreta acaba de matar a una araña y se dispone a disparar a la siguiente. Este sistema de partículas se ha obtenido del mismo pack que el anterior.

- **Explosión**: este sistema de partículas es usado cuando los monstruos consiguen destruir las torretas.

Figura 78: Representación del sistema de partículas "explosión"

Cuando la vida de la torreta llega a 0 se instancia, haciendo que el jugador se percate de su destrucción cuando está en pantalla.

Este sistema de partículas se ha obtenido a partir del pack del Asset Store llamado "Simple particle pack" del publicador "Unity Technologies".

- **Polvo:** este sistema se partículas se instancia una vez que uno de los monstruos acaban con las vallas. Simula una polvareda de humo que se va extendiendo poco a poco hasta que consigue difuminarse.

Figura 79: Representación del sistema de partículas "polvo"

Este sistema de partículas se ha obtenido también a partir del sistema de partículas anterior aunque con ciertas modificaciones, para conseguir que se expanda más y durante más tiempo.

4.2 Implementación del proyecto

4.2.1 Lenguaje de programación:

Como se ha comentado anteriormente en este documento, el lenguaje que se ha empleado para la implementación del proyecto ha sido JavaScript.

El lenguaje de programación JavaScript es utilizado fundamentalmente para la creación de páginas web dinámicas. Estas páginas incluyen efectos como aparición o desaparición de texto, animaciones, activación de acciones por medio de pulsación de botones y mensajes para avisar al usuario por medio de ventanas.

Desde el punto de vista técnico, JavaScript es un lenguaje interpretado, lo que quiere decir que no es preciso que los programas deban compilarse para su ejecución, es decir, que estos programas pueden ser probados en cualquier tipo de navegador, sin ser necesarios otros procesos. El nombre del programa JavaScript puede dar lugar a la confusión de que está muy relacionado con el lenguaje Java, pero sin embargo desde el punto de vista legal JavaScript está registrado por la empresa Sum Microsystems, como se puede ver en http://www.sun.com/suntrademarks/ y no está relacionado en forma directa con Java.

La herramienta Unity puede utilizarse con 3 lenguajes de programación: Boo (la cual es una referencia a Python), C# y Javascript (conocido también como Unityscript).
El más sencillo de los 3 es el último y es el que presenta mayores semejanzas con el lenguaje Java.

Indicaremos las 3 ventajas más importantes de Javascript para usarlo con Unity:

1) **Flexibilidad:** cuando en Javascript se declara una variable no es preciso declarar su tipo al utilizarla, cuando lo normal en la mayoría de los lenguajes de programación es declarar el tipo de variable que se va a usar.

2) **Es fácilmente manejable y entendible:** al no ser tan riguroso como ocurre con C#, es más fácilmente entendible, denominándolo a esto Tipado Dinámico.

3) **Una Guía de programación con mayor soporte:** En la guía de Unity Scripting Reference, la documentación es mucho más completa que para los lenguajes restantes usados por Unity, siendo JavaScript el utilizado como lenguaje predeterminado:
https://docs.unity3d.com/Documentation/ScriptReference/index.html

   Hay una gran cantidad de motores gráficos que utilizan lenguaje script como por ejemplo Hero Engine y Unreal Engine. Ambos utilizan los lenguajes integrados de scripting HeroScript y UnrealScript respectivamente y que ambos están basados en Javascript o equivalente. Algunas aplicaciones como Flah también incluyen lenguajes de script (ActionScript) similar a JavaScript.

Otras razones para el uso de JavaScript pueden ser que este lenguaje tiene una estructura muy sólida, con la necesidad de no tener grandes conocimientos para su iniciación y además resulta bastante conocido por la comunidad de programadores.

Es preciso indicar que Unity no utiliza un JavaScript puro, sino que al compilarse lo hace más rápidamente y ha sido extendido para que contenga funciones propias para el desarrollo de un juego.

Podemos resaltar que, al contrario que con JavaScript estándar, en Unity si tenemos que declarar el tipo de variable como tradicionalmente lo hace C y C++.

Por ejemplo, para declarar la variable **numero** de tipo int se haría:

```
var numero : int;
numero = 8;
```

Y si la quisiéramos inicializar en la declaración, sería:

```
var numero : int = 8;
```

Esto quiere decir que la palabra "**var**" reservada se utiliza para indicar que se va a declarar una nueva variable, después vendrá el nombre que le demos a ésta seguida de dos puntos y el tipo de variable declarada.

Por otra parte diremos que la API de Unity está orientada a objetos y tiene algo más de 100 clases, de las que aproximadamente la tercera parte se relacionan entre sí en términos de herencia y las demás son grupos de funciones auxiliares (clase tiempo, de funciones matemáticas, etc…) o tiene posibilidad de utilizaciones especiales que no se prevén en las clases troncales (las relativas a la GUI o por ejemplo las utilizadas para conectarse a un servidor).

4.2.2 Estructura básica de un script en Unity

A continuación vamos a explicar la estructura de un script en Unity. Esta estructura es independiente del lenguaje que elijas (**C#**, **Javascript** o **Boo**).

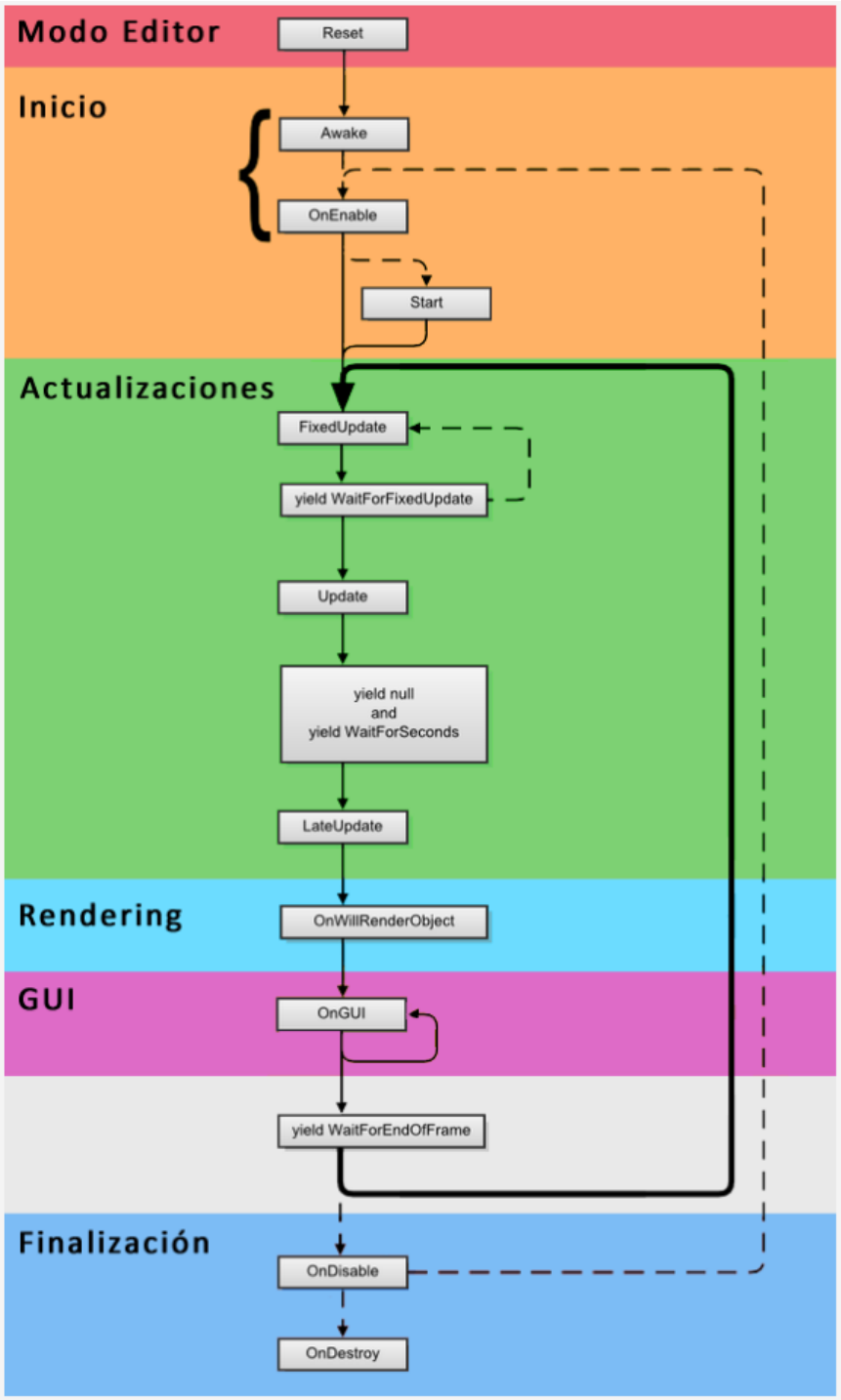


Figura 80: Diagrama que explica la estructura de un script en Unity 3D [36]

[36] *Imagen obtenida desde* http://www.widget-101.com/juegos/unity-3d/introduccion-a-unity-3d/ *en Junio de 2014*

- **Reset:** Método invocado cuando se añade un script a un **GameObject** o cuando el componente se ha reseteado por el diseñador por medio del inspector. Resulta especialmente útil cuando se inicializan valores por defecto.

- **Awake:** Se usa para inicializar variables o estados del juego antes del comienzo del mismo y se invoca una única vez, no volviendo a invocarse más durante todo el tiempo de vida del script. Se puede considerar como constructor de una clase.

- **Start:** Igual que el anterior, solo se invoca una sola vez durante todo el tiempo de vida del script. La diferencia fundamental entre este y el anterior, es que éste solo se invocaría cuando el GameObject esté activo, con lo que se consigue retrasar hasta ser realmente necesario cualquier código de inicialización

- **OnEnabled:** Se invoca cada vez que se habilita el **GameObject**.

- **Update:** Es invocado una vez por frame, siendo necesario que el **GameObject** esté habilitado.

- **FixedUpdate:** Es llamado cada vez sea necesario realizar operaciones físicas sobre el objeto y siempre que el GameObject se encuentra activo. Un ejemplo de utilización puede ser aplicar una fuerza a un objeto rígido; en este caso debe utilizarse FixedUpdate en vez de Update.

- **LateUpdate:** Al igual que Update, este método se invocará cuando el GameObject esté habilitado y una vez por frame, siendo llamado después de Update y como una actualización posterior del objeto.

- **OnWillRenderObject:** Este método se invocará cuando el GameObject esté habilitado y una vez por frame. Además el GameObject debe contener un renderizador (Renderer) y el objeto debe estar dentro del campo de visión de la cámara.

- **OnGUI:** Este método es llamado varias veces por frame, debiendo indicar los elementos que componen la interfaz gráfica.

- **OnDisable:** Es llamado cuando se deshabilita el objeto.

- **OnDestroy:** Es llamado cuando se destruye el objeto y siempre y cuando dicho objeto haya permanecido activo.

Podemos indicar que, independientemente del lenguaje utilizado, cuando creamos un script aparecerán por defecto las funciones Start() y Update() y el diseñador rellenará estas funciones si es conveniente.

- **Start()** se usa para inicializar variables porque el código que incluye será el primero en ejecutarse.

- **Update()** se usa para introducir el código preciso que será ejecutado por cada fotograma del videojuego, por lo que resulta el punto más importante para el flujo del mismo [19]

4.2.3 Partes de código más destacables

En este apartado vamos a destacar determinadas partes del código de algunos scripts que es necesario explicar para entender correctamente su funcionamiento.
Concretamente nos vamos a centrar, dentro de la IA de los enemigos, en cuál es el papel que desempeñan los nodos de cara al recorrido que llevan a cabo para llegar a la aldea, así como el uso de Raycast para la detección de obstáculos.

4.2.3.1 IA de los enemigos: funcionamiento de los Nodos.

En el **apartado 4.1.2** se ha explicado a través de diversos diagramas de flujo los diferentes estados por los que va atravesando la IA de los enemigos. En cambio, no se detalla en los diagramas cómo se comporta de cara a dirigirse hacia un determinado nodo destino. Esto es debido a que el paso de un nodo a otro se ejecuta en un código aparte dentro del script y su ejecución es independiente de los estados por los que puede pasar la IA del enemigo.

En primer lugar, en el script se definirá una matriz, llamada "**matrizPivotes**", en forma de array de 6 filas por 5 columnas. Éstas se corresponderán a la forma en que espacialmente hemos dividido los nodos "**pivote**" por el escenario (**Ver Figura 65: Mapa del juego dividido en filas y columnas**) así como una variable de tipo Vector3 llamada "**posicionDestino**" que es capaz de almacenar un punto en el espacio. Este punto corresponderá a la posición del nodo destino.

También se definirá la variable "**objetoMovil**" como un componente de tipo "**NavMeshAgent**" que guardará en este caso el objeto o personaje que se desplazará por el mapa. En nuestro caso los Nav Mesh Agent serán los monstruos.

Definiremos también propiedades del Nav Mesh Agent como es la velocidad con la que se desplazará en el estado "Patrulla" o en el estado "Perseguir":

```
/* Variables para el Nav Mesh */
// objetoMovil hace referencia a nuestro personaje enemigo (que se va a mover por el
mapa inteligentemente)
var objetoMovil = gameObject.GetComponent.<NavMeshAgent>();

// Variables que controlan la velocidad del Nav Mesh Agent
public var speedPatrulleo : float;
public var speedPerseguir : float;
```

```
/* Variable para la matriz de pivotes */
var matrizPivotes: Array[]; // Array de arrays (osea una matriz)
var fila : int; // fila de pivotes (va en orden de una en una)

/* Variable global que guarda la actual posicon del PivoteDestino hacia el que se dirige
el monstruo */
var posicionDestino : Vector3;
```

Para inicializar los valores de la matriz o acceder a ellos, se crearán antes dos métodos:

```
/* Funciones para la matriz de pivotes */
function setMatriz(fila:int, array:Array){
  matrizPivotes[fila] = array;
  print("Fila "+fila+" añadida :"+matrizPivotes[fila]);
}

function getMatriz(fil:int, col:int){
var array : Array = matrizPivotes[fil];
return array[col];
}
```

La inicialización tendrá lugar en la función **Start()** a través de las siguientes líneas de código:

```
// Inicializamos la matriz de pivotes
matrizPivotes = new Array[6]; // vamos a tener una matriz 6x5

setMatriz(0, new Array(GameObject.Find("Pivote 00"), GameObject.Find("Pivote 01"),
GameObject.Find("Pivote 02"), GameObject.Find("Pivote 03"), GameObject.Find("Pivote
04")));
setMatriz(1, new Array(GameObject.Find("Pivote 10"), GameObject.Find("Pivote 11"),
GameObject.Find("Pivote 12"), GameObject.Find("Pivote 13"), GameObject.Find("Pivote
14")));
setMatriz(2, new Array(GameObject.Find("Pivote 20"), GameObject.Find("Pivote 21"),
GameObject.Find("Pivote 22"), GameObject.Find("Pivote 23"), GameObject.Find("Pivote
24")));
setMatriz(3, new Array(GameObject.Find("Pivote 30"), GameObject.Find("Pivote 31"),
GameObject.Find("Pivote 32"), GameObject.Find("Pivote 33"), GameObject.Find("Pivote
34")));
setMatriz(4, new Array(GameObject.Find("Pivote 40"), GameObject.Find("Pivote 41"),
GameObject.Find("Pivote 42"), GameObject.Find("Pivote 43"), GameObject.Find("Pivote
44")));
setMatriz(5,  new  Array(GameObject.Find("Pivote  c1"),  GameObject.Find("Pivote  c2"),
GameObject.Find("Pivote  42"),  GameObject.Find("Pivote  c3"),  GameObject.Find("Pivote
c4")));
```

Todos los monstruos o enemigos una vez que aparecen en el mapa, se desplazarán partiendo de la fila 0. Respecto al nodo destino qué elegirán dentro de la fila 0, este será totalmente aleatorio. Por tanto, dentro de la función **Start()** se definirán también las siguientes líneas:

```
// Los monstruos comienzan en la fila 0
fila = 0;

// Nada más arrancar, el monstruo elegirá un destino
posicionDestino = GetRandom().transform.position;
objetoMovil.destination = posicionDestino;
```

El método **GetRandom()** se definirá de la siguiente forma:

```
/*  Función que devuelve una variable Transform (propiedades del pivoteDestino) para
conseguir obtener aleatoriamente la posición de un pivoteDestino, siguiendo un orden:
filas de arriba a abajo, y columnas aleatorias
*/

function GetRandom() : GameObject{
    var col = Random.Range(0,4);
    var pivoteDestino : GameObject = getMatriz(fila, col);
    //print("PivoteDestino ="+pivoteDestino);
    fila++;
    return pivoteDestino;
}
```

Por último, con estas líneas de código y con los métodos anteriormente definidos conseguimos que los enemigos vayan moviéndose a lo largo del mapa de forma totalmente autónoma, independientemente de los estados por los que pasa en los diagramas de flujo.

```
/* Codigo para el Nav Mesh:
    se ejecutara cuando el monstruo llegue al pivote correspondiente
*/
function OnTriggerEnter (other : Collider) {
    posicionDestino = GetRandom().transform.position;
    objetoMovil.destination = posicionDestino;
}
```

Como se explicó anteriormente, los **Triggers** no son otra cosa que disparadores o activadores de eventos. En este caso, los nodos pivote serán **Triggers** los cuales al ser atravesados por el Nav Mesh Agent ejecutarán el método **OnTriggerEnter** que permitirá, llamando a la función **GetRandom(),** elegir un nuevo pivote destino hacia el cual se dirigirá el Nav Mesh Agent. Así es como los monstruos se irán desplazando de fila en fila, siguiendo un orden pero por caminos aleatorios hasta llegar a la última fila correspondiente a la aldea.

Por último, destacar que cuando la IA de los monstruos entra dentro del estado "**Perseguir**", la nueva posición destino hacía en el que se dirigirá el Nav Mesh Agent ya no serán los nodos pivote, sino la posición del Player, permitiendo de esta forma que los monstruos persigan al Player para atacarle. Además se modificarán los parámetros de velocidad del Nav Mesh Agent definidos en los atributos según el estado en que nos encontremos. En caso de volver al estado "**Patrulla**", la posición destino volvería a ser el nodo pivote al que se dirigía el monstruo antes de cambiar de estado.

4.2.3.2 IA de los enemigos: Raycast de los enemigos

En el apartado anterior se ha explicado detalladamente como se hace uso del componente Nav Mesh Agent para que los enemigos vayan desplazándose por el camino. En cambio hay que destacar un detalle importante en el funcionamiento de la IA. Mientras el monstruo se encuentra desplazándose hacia un pivote, este se encuentra rastreando permanentemente a través de **Raycast** qué es lo que se encuentra en frente suya. Es el sistema de "visión" que utilizamos para que los monstruos no pasen de largo si por ejemplo encuentran por el camino una valla o una torreta.

Para entender que es un **Raycast**, decir que éste se trata de un rayo que se lanza desde un determinado punto hacia todos los **colliders** en la escena.

La función utilizada es:

static function Raycast (origin : Vector3, direction : Vector3, distance : float, layerMask : int) : boolean

Devuelve true cuando el rayo intersecta algún **collider**. A continuación vamos a describir cada uno de los parámetros:

```
origin        El punto inicial del rayo en coordenadas globales.
direction     La dirección del rayo.
distance      La longitud o fuerza del rayo.
layerMask     Una máscara de distribución (Layer mask) que se usa para ignorar
              selectivamente colliders cuando se proyecta un rayo.
```

A continuación se muestra una parte de código del estado **Perseguir** de nuestro proyecto para que se pueda apreciar cómo funciona:

```
case AIState.Perseguir:
  if(distanciaPlayer < searchRange){
   SendMessage("SetSpeed", 2); //animation correr
   objetoMovil.speed = speedPerseguir;
   objetoMovil.destination = player.position;

// A lo largo del desplazamiento con Nav Mesh, nuestro enemigo analizará con un Raycast
lo que se encuentra en frete suya

if(Physics.Raycast(transform.position, transform.forward, hit, longitudRayo )){

 if(    hit.collider.gameObject.tag=="Torreta" ||
        hit.collider.gameObject.tag == "VallaAldea" ||
        hit.collider.gameObject.tag=="VallaLaberinto"){
  //almacenamos en "objetoDetectado" el objeto que tiene como tag "Obstaculo"
  objetoDetectado = hit.collider.gameObject;//transform;
  //obtenemos de este objeto, su script
  scriptObstaculo = objetoDetectado.GetComponent(VidaObstaculo);
  state = AIState.Romper;

 }

//almacenamos en "objetoDetectado" el objeto que tiene como tag "Obstaculo"
  if(hit.collider.gameObject.tag=="Casa"){
  //almacenamos en "objetoDetectado" el objeto que tiene como tag "Casa"
  objetoDetectado = hit.collider.gameObject; //tranform es una propiedad que guarda las
caracteristicas del objeto
  state = AIState.Romper;
  }

}
```

En este fragmento apreciamos cómo a la vez que el monstruo se encuentra persiguiendo al Player, este va analizando a través de Raycast si encuentra un obstáculo qué destruir. Como podemos observar en el código, el comportamiento es distinto si detecta las casas de la aldea a si detecta el resto de objetos destruibles, y por ello se localizan en condicionales distintas.

El Raycast también es usado en el estado Patrulla para detectar los objetos destruibles mientras los monstruos se dirigen hacia los nodos pivote.

4.2.3.3 Lectura de ficheros XML en Unity

Usar ficheros XML es muy útil ya que nos permite generar ficheros que, al abrirlos un humano, este lo puede comprender, no está en binario y tienen una estructura fácilmente comprensible para nosotros que nos permite manipularlo a mano sin problemas.

En el caso de este proyecto, una de las prioridades era conseguir que una persona sin conocimientos técnicos pudiera modificar o insertar las cuestiones que posteriormente se le formularán al usuario sin dificultad alguna. Es por esto que se ha optado por el uso de un fichero XML con una estructura sencilla y fácil de entender para cualquier persona.

La tecnología .NET introduce una serie de APIs XML basadas en estándares como DOM, XPath, XSD, XSLT y SOAP. Tales APIs están compuestas por una serie de clases que pertenecen a varios namespaces (siento el contenedor de todos System.Xml) y que hacen más fácil, flexible e intuitiva la programación de aplicaciones con soporte XML.

En la MSDN Library podremos encontrar información acerca de cómo funciona System.Xml que usa Unity 3D ya que usa la .NET Framework 2.0
http://msdn.microsoft.com/es-es/library/System.Xml(v=vs.80).aspx

También se hará uso de la librería System.IO de la plataforma .NET que nos proporciona una serie de clases que nos permiten trabajar con el sistema de archivos.
En este caso se utilizará para la entrada de datos del fichero XML.

4.3 Problemas y soluciones aplicadas

4.3.1 Memoria RAM llena:

Al principio, en el juego se separó el menú principal de la pantalla de juego a través de 2 escenas independientes.

En Unity una escena puede ser cualquier parte del juego, desde el menú de inicio como un nivel o área del juego. Una escena es como un lienzo en blanco sobre el que dibujar cada parte del juego usando las herramientas de Unity.

Por tanto, siguiendo la lógica se separó la escena de la pantalla principal donde se mostraba el título del juego, de la escena del juego propiamente dicha.

El problema surgió que al enlazar la pantalla principal con la del juego, éste dejaba la pantalla totalmente en negro durante unos segundos hasta que finalmente se salía bruscamente del mismo. Estas pruebas fueron realizadas con el móvil Samsung Galaxy 2 y con el tablet Samsung Galaxy Tab 2. En ambos casos se observó el mismo error.

Analizando el caso se llegó a la conclusión de que podía ser un problema de falta de memoria RAM, ya que tendría que cargar objetos en la pantalla principal que posteriormente tendrían que ser destruidos para volver a cargarlos de nuevo en otra escena distinta junto con otro montón de objetos.

Es por esto por lo que se optó por una mayor optimización, estructurando la vista de jerarquía de una forma totalmente distinta a como la teníamos en principio.

Finalmente se optó por dividir la jerarquía en:

- **Pantalla principal**: que reuniría los objetos propios de la pantalla de presentación del juego.

- **Pantalla de juego**: que reunirá todos los objetos propios del juego en sí.

- **Objetos comunes**: que reúne aquellos objetos que serán utilizados tanto en la pantalla principal como en la pantalla de juego

Todos estos objetos serán cargados en una única escena en vez de 2 escenas diferentes. Así pues, los objetos comunes se encontrarán activos en todo momento, ya que son utilizados en ambas escenas, mientras que los objetos de la pantalla principal permanecerán activos siempre que el juego se dirija a la pantalla principal, desactivando todos los demás objetos de la pantalla de juego. Igualmente, cuando los objetos de la pantalla de juego se encuentren activos, los objetos de la pantalla principal se desactivarán. Con esta estrategia conseguimos optimizar mejor el uso de la memoria, ya que todos los objetos de juego se cargarán en memoria una única vez, en una misma escena. Tan solo los desactivaremos o activaremos cuando vayamos a hacer uso de ellos.

4.3.2 Bajo frame rate: perspectiva de la cámara

Antes que nada es importante explicar los 2 tipos de perspectiva más importantes que existen de cara a como se representan los objetos en pantalla.

Los 2 tipos de proyección principales que vale la pena considerar para juegos esencialmente en 2D: la proyección ortográfica (1), y la proyección en perspectiva (2).

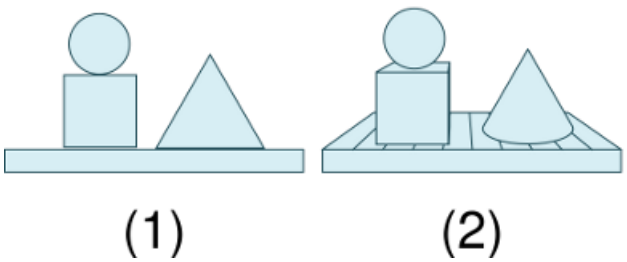


Figura 81: Representación de los tipos de proyecciones en 2D

En la primera, siempre veo la cara frontal del objeto, y nada más. En el segundo caso, en cambio, los objetos tienen una profundidad que cambia dependiendo desde donde las mire.

Al principio cuando se estaba iniciando el desarrollo del juego se optó por una cámara de proyección en perspectiva:

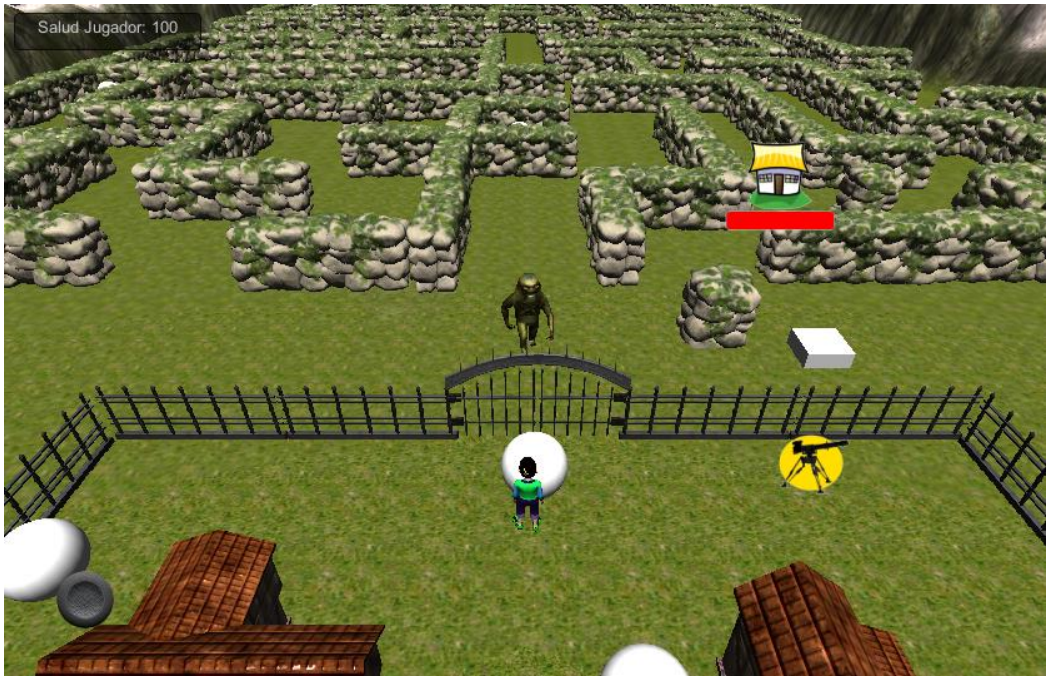


Figura 82: Imagen del juego en desarrollo con cámara de proyección en perspectiva

En cambio, en las pruebas realizadas tanto en un móvil Samsung Galaxy 2 como en un Tablet Samsung Galaxy Tab 2, se pudo observar que el rendimiento del juego era bastante mediocre. Esta falta de rendimiento se vio reflejada en un frame rate bastante pobre por lo que no se conseguía la suavidad deseada en el movimiento de la cámara ni en el desplazamiento del personaje.

Tras investigar, se estuvo analizando la importancia del número de **Draw Calls**.
Uno de los aspectos más importantes a tener en cuenta cuando pensamos en optimizar el rendimiento de nuestro juego es el concepto de **draw calls** y el número de triángulos representados en pantalla.
Este es un elemento que podemos determinar con facilidad en la ventana de juego (**game view**), activando la solapa de **stats**. En ella, aparece el número de **draw calls** entre la diversa información al respecto del rendimiento de nuestro juego en ejecución.

Echando mano a la documentación oficial de Unity3D, encontramos que el número de **draw calls** son el total de mallas dibujadas tras el proceso de **batching**. **Batching** es el proceso donde el motor intenta combinar el renderizado de múltiples objetos en un único **draw call** para reducir la sobrecarga de la CPU.

Para una mayor optimización, se estuvo estudiando la cantidad de **drawcalls** que se dibujaban en pantalla si optábamos por usar una **cámara con proyección en perspectiva**. En cambio, si elegíamos hacer uso de una **cámara con proyección ortogonal**, el número de **drawcalls** se reducía drásticamente, permitiendo darle al juego otro aspecto en cuanto a la forma de verlo y sobretodo se conseguía ganar un buen rendimiento.

De esta forma la vista del juego ha quedado situando la cámara más arriba para poder abarcar más el escenario y utilizando una perspectiva que recuerda a los juegos 2D.

A este tipo de juegos como el de nuestro proyecto se los conoce como juegos con **perspectiva top-down** (*perspectiva arriba-abajo*), también conocida como **vista ojo de ave**, **vista elevada** o **vista de helicóptero**, los cuales tienen un ángulo de cámara que muestra al jugador y al área circundante desde arriba.

4.3.3 NavMesh y sus problemas

Una de las dificultades que nos encontramos con el uso de NavMesh como sistema de navegación es que en este sistema no se calculan los caminos en tiempo real. Es decir, que todo se pre-calcula para que en runtime se pueda procesar rápidamente. Esto quiere decir que para encontrar un camino para llegar a un destino, Unity ya debe conocer los obstáculos que tendrá que librar de por medio para llegar a este punto.
Esto es debido, como se ha explicado más arriba, a que Unity previamente tiene que crear una malla para poder separar los obstáculos del terreno.

Si el obstáculo no estuviere previsto y se colocase en un lugar cualquiera del mapa, el Nav Mesh Agent lo ignorará y lo atravesará como si en realidad no existiese. La solución a este problema fue incluir en la IA de los enemigos un código especial que fuese ejecutado en el momento en que los enemigos detectasen a través de Raycast un obstáculo nuevo. Estos obstáculos serían bien una torreta o una valla colocadas por el jugador. El comportamiento del monstruo será acercarse a dicho obstáculo y atacarlo hasta destruirlo.

El código que incluimos para este propósito se extrajo del método MoveTowards (position : Vector3), del script AIRobot.js perteneciente a un tutorial online de MrJocyf.

Haciendo Enemigos en Unity3D - Parte 1: https://www.youtube.com/watch?v=kiFG3vZEzt8
Haciendo Enemigos en Unity3D - Parte 2: https://www.youtube.com/watch?v=duff7mR_YwQ
Enlace de descarga del proyecto de ejemplo:
http://www.jocyf.com/Files/Tutorials/ScreenCast_Unity3D_GeneralVars_Enemigos_4.0.rar

El código en cuestión es el siguiente:

```
/*** CODIGO DE AIROBOT: MEVETOWARDS Y ROTATETOWARDS ***/
//* Moverse hacia un punto rotando a la vez hacia el blanco.
// Este código es utilizado para dirigirme hacia un obstáculo y romperlo, puesto que con
el NavMesh, el personaje enemigo atraviesa los obstáculos y no respeta las colisiones

function MoveTowards (position : Vector3){ // posición estimada a la que tiene que irse
    var direction : Vector3 = position - transform.position; // posición del enemigo -
    la posición de destino
    direction.y = 0;
    if (direction.magnitude < 0.5){
        SendMessage("SetSpeed", 0.0);
        return;
    }



    // ---------- Rotacion -------
    // Rotate towards the player
    //- Slerp: interpola para que vaya rotando poco a poco hacia su destino
    //- transform.rotation: rotación actual que tiene le enemigo (sea donde sea donde
este mirando)
    //- Quaternion.LookRotation(direction): Busca la rotación necesaria que tiene que
tener el enemigo cuando está encarando en esa dirección
    transform.rotation = Quaternion.Slerp (transform.rotation, //rotacion inicial
    Quaternion.LookRotation(direction), rotationSpeed * Time.deltaTime);
       // rotacion final                    velocidad de rotacion

    transform.eulerAngles = Vector3(0, transform.eulerAngles.y, 0); // aplico esa
rotacion solo en el eje Y


    // ---------- Movimiento -------
    // Calcula el movimiento que se genera para el CharacterController
    // Modify speed so we slow down when we are not facing the player
    var forward : Vector3 = transform.TransformDirection(Vector3.forward);
    var speedModifier : float = Vector3.Dot(forward, direction.normalized); //mira a qué
velocidad tiene que ir haciendo un producto entre la direccion "forward" y la "direccion
normalizada" cuya longitud no sea 1
    speedModifier = Mathf.Clamp01(speedModifier); //no permite que el valor pase de 0,1

    // Move the character
    direction = forward * speed * speedModifier;
    var MyController : CharacterController = GetComponent (typeof(CharacterController))
as CharacterController;
    MyController.SimpleMove(direction);

    SendMessage("SetSpeed", speed * speedModifier,
SendMessageOptions.DontRequireReceiver);
}
```

Capítulo 5: Evaluación y resultados

5.1 Evaluación

Para poder evaluar esta aplicación y sacar conclusiones sobre cuáles son los puntos a favor y los puntos en contra, así como obtener conclusiones sobre qué se debe mejorar en un futuro, se han aplicado métodos cuantitativos, realizando una encuesta a 5 personas cercanas a mi entorno social.

Las personas participantes han resuelto dos tipos de encuesta:

La primera de ellas se realiza antes de probar el juego y la segunda después de jugar.
En la primera encuesta, nos aseguramos que la persona que va a evaluar nuestro juego cumple con las características propias del perfil que buscamos. Este perfil se basa a grandes rasgos, de alguien que ya tenga experiencia en el uso de videojuegos y que realmente le guste jugar.

En la segunda encuesta se trata de evaluar los diferentes aspectos que componen el juego y que pueden ayudarnos a conocer lo que piensa un usuario acerca de nuestro juego, si se divierte, si aprende y si en general la experiencia es satisfactoria.

Para poder responder a la segunda encuesta, es necesario que la puntuación sea positiva en la primera, ya que si no, puede ocurrir que los resultados no se ajusten al perfil deseado.

La primera encuesta se basa en un total de 6 preguntas y pueden puntuar con un valor de aprobación o desaprobación basado en una enumeración del 0 al 5 siendo:

0 – nada
1 – muy poco
2 – poco
3 – habitualmente
4 – mucho
5 – muchísimo

Encuesta de preguntas antes de jugar:

1. Me gustan los videojuegos
2. Soy una persona que juego habitualmente a los videojuegos
3. Suelo usar el móvil o Tablet para jugar
4. He jugado a juegos educativos
5. Me gustan los juegos educativos
6. Aprendo bastantes cosas con juegos educativos

Aquí se muestra la primera tabla de la encuesta:

Preguntas	Evaluación				
	Persona encuestada				
	P1	P2	P3	P4	P5
Me gustan los videojuegos					
Soy una persona que juego habitualmente a los videojuegos					
Suelo usar el móvil o Tablet para jugar					
He jugado a juegos educativos					
Me gustan los juegos educativos					
Aprendo bastantes cosas con juegos educativos					
Media					
Desviación típica					

Tabla 29: Primera tabla modelo de encuesta

La segunda encuesta se basa en un total de 12 preguntas y pueden puntuar con un valor basado en una enumeración del 0 al 5, siendo 0 si no estás en absoluto de acuerdo, o 5 si estás totalmente de acuerdo:

0 – en absoluto
1 – muy poco
2 – poco
3 – normal
4 – de acuerdo
5 – totalmente de acuerdo

Encuesta de preguntas después de jugar

1. Me parece bueno el planteamiento del juego
2. Me parece un juego innovador
3. Me ha gustado y estoy satisfecho con la experiencia realizada con el juego
4. Me parece correcta la interfaz del juego
5. Me parece correcta su jugabilidad
6. Me ha parecido intuitivo su uso
7. Me parece correcta su dificultad
8. Me ha parecido entretenido
9. Me ha parecido adictivo
10. Creo haber aumentado mi aprendizaje de las materias que se han tratado
11. Me gustaría evaluar más mis conocimientos haciendo uso de este juego
12. La utilización de este juego me ha motivado a seguir intentando resolverlo

Aquí se muestra la segunda tabla de la encuesta:

Preguntas	Evaluación						
	Persona encuestada					Resultados estadísticos	
	P1	P2	P3	P4	P5	Media	Desv. Típica
Me parece bueno el planteamiento del juego							
Me parece un juego innovador							
Me ha gustado y estoy satisfecho con la experiencia realizada con el juego							
Me parece correcta la interfaz del juego							
Me parece correcta su jugabilidad							
Me ha parecido intuitivo su uso							
Me parece correcta su dificultad							
Me ha parecido entretenido							
Me ha parecido adictivo							
Creo haber aumentado mi aprendizaje de las materias que se han tratado							
Me gustaría evaluar más mis conocimientos haciendo uso de este juego							
La utilización de este juego me ha motivado a seguir intentando resolverlo							

Además de la tabla, al encuestado se le formulan las siguientes cuestiones:

Por favor, indique los tres aspectos de la experiencia que considera más positivos
Por favor, indique los tres aspectos de la experiencia que considera más negativos

Tabla 30: Segunda tabla modelo de encuesta

5.2 Resultados

Ahora se detallan los resultados obtenidos de las encuestas tanto de las tablas 4.1. y 4.2 como de las 2 últimas preguntas. También se ha calculado la media y la desviación típica de las muestras obtenidas. En la tabla representamos a cada una de las personas encuestadas desde P1 (primera persona encuestada) hasta P5 (última persona encuestada). Hay que tener en cuenta que los resultados de las tablas 4.4 y 4.6 se han calculado en base a un valor máximo de 5.

Resultados de la encuesta realizada antes de jugar:

Preguntas	Evaluación				
	Persona encuestada				
	P1	P2	P3	P4	P5
Me gustan los videojuegos	5	3	2	4	4
Soy una persona que juego habitualmente a los videojuegos	5	3	2	2	5
Suelo usar el móvil o Tablet para jugar	4	4	3	5	2
He jugado a juegos educativos	4	4	4	3	3
Me gustan los juegos educativos	4	4	4	2	2
Aprendo bastantes cosas con juegos educativos	4	3	3	3	2
Media	**4.3**	**3.5**	**3**	**3.16**	**3**
Desviación típica	**0.46**	**0.5**	**0.81**	**1.06**	**1.15**

Tabla 31: Primera tabla de resultados de la encuesta

Resultados de la encuesta realizada después de jugar:

Preguntas	Evaluación						
	Persona encuestada					Resultados estadísticos	
	P1	P2	P3	P4	P5	**Media**	**Desv. Típica**
Me parece bueno el planteamiento del juego	5	5	5	4	2	**4.2**	**1.16**
Me parece un juego innovador	4	3	3	5	2	**3.4**	**1.02**
Me ha gustado y estoy satisfecho con la experiencia realizada con el juego	4	5	4	4	2	**3.8**	**0.98**
Me parece correcta la interfaz del juego	5	4	4	3	3	**3.8**	**0.74**
Me parece correcta su jugabilidad	4	4	5	4	2	**3.8**	**0.98**
Me ha parecido intuitivo su uso	5	5	5	5	5	**5**	**0**
Me parece correcta su dificultad	3	4	3	2	4	**3.2**	**0.74**
Me ha parecido entretenido	4	4	4	3	3	**3.6**	**0.5**
Me ha parecido adictivo	4	3	4	2	2	**3**	**0.89**
Creo haber aumentado mi aprendizaje de las materias que se han tratado	3	4	3	2	1	**2.6**	**1.02**
Me gustaría evaluar más mis conocimientos haciendo uso de este juego	5	3	5	1	2	**3.2**	**1.6**
La utilización de este juego me ha motivado a seguir intentando resolverlo	4	3	5	1	2	**3**	**1.41**

Tabla 32: Segunda tabla de resultados de la encuesta

Observando los resultados de esta tabla, puesto que la media de todas las preguntas es superior a 2.5 (que representaría el aprobado) podemos concluir que la apreciación del juego por parte de los encuestados se considera satisfactoria.

Si profundizamos más en los valores obtenidos, podemos decir que los puntos de menos satisfacción por parte de los usuarios que lo han probado y han jugado a él repetidas veces ha sido su dificultad, para algunos demasiado fácil, y su necesidad de añadir más elementos que lo vuelvan mucho más adictivo y con más variedad de armas a la hora de acabar con las hordas de enemigos. Aun así, los usuarios aprueban con nota el juego, considerándolo entretenido, innovador y con un planteamiento más que correcto.

En cuanto a su parte educativa, decir que la mayor parte de los usuarios afirman que el juego es bastante bueno para evaluar nuestros conocimientos, pero no tan bueno para aprender conceptos nuevos, ya que se trata de preguntas tipo test y no da opción a conocer una explicación detallada de porqué una respuesta es o no correcta.

Los valores de las desviaciones típicas nos muestran que la media en cada pregunta es bastante representativa de las calificaciones que dan los encuestados, puesto que en bastantes casos, no llega a la unidad.

Cuestiones adicionales:

Por favor, indique los tres aspectos de la experiencia que considera más **positivos**:

- Encuestado nº 1: Entretenido, interfaz atractiva y muy fácil de manejar.
- Encuestado nº 2: Ameno, original en su parte educativa y fácil de utilizar.
- Encuestado nº 3: Hace pensar, es muy intuitivo y engancha bastante.
- Encuestado nº 4: Distinto a lo ya visto, fácil de usar y rápido para jugar en sesiones cortas
- Encuestado nº 5: Intuitivo de manejar, dinámica interesante y entretenido.

Por favor, indique los tres aspectos de la experiencia que considera más **negativos**:

- Encuestado nº 1: Necesita más dificultad, escasez de niveles, se repiten las preguntas.
- Encuestado nº 2: Dinámica muy repetida en otros juegos, necesita incluir funciones online como compartir puntuación, se hace un poco repetitivo.
- Encuestado nº 3: Dificultad bastante fácil, añadir más monstruos, poder recuperar la sesión anterior.
- Encuestado nº 4: Necesario incluir una pantalla que indique a qué nivel pasas, falta añadir puntuación por pregunta acertada para motivar más al jugador y que no se repitan, falta añadir más niveles.
- Encuestado nº 5: Jugabilidad un poco escasa, muy facilón una vez que se ha jugado varias veces, se aprenden pocas cosas debido a que solo sirve para evaluar conocimientos.

Como conclusiones de la encuesta, podemos decir que el juego es aprobado por las personas que han tenido ocasión de jugarlo. En líneas generales, lo consideran un juego atractivo y entretenido, con buen planteamiento y original en su forma de mezclar la acción con lo educativo.

Por el contrario, como aspectos a mejorar habría que destacar que se necesita añadir más niveles o escenarios que permitan que el juego no se haga tan repetitivo, enemigos nuevos, y la posibilidad de aumentar su dificultad para motivar más a los jugadores. También sería importante corregir ciertos aspectos en el interfaz como añadir información de cuando empezamos un nuevo nivel, incluir estadísticas de las preguntas acertadas y añadir funciones online que lo hagan más sociable.

Capítulo 6: Conclusiones y trabajos futuros

6.1 Conclusiones:

Como conclusión, se puede decir que la aplicación ha sido desarrollada con éxito y las personas encuestadas que han tenido la oportunidad de probarlo, han evaluado el juego como original y bastante atractivo, además de haber tenido la oportunidad de evaluar sus conocimientos sobre diversas materias.

De esta forma, se ha obtenido un juego dinámico y llamativo que busca fundamentalmente entretener al usuario en sus ratos libres, a la vez que examina su nivel de conocimiento de las materias que el docente desee establecer. Además, través del ensayo-error, el usuario aprende a responder correctamente a una pregunta en caso de no conocer la respuesta, por lo cual, el juego cumple con el objetivo educativo y evaluativo deseado.

En lo que respecta a su dinámica, deja de convertirse en un simple juego de preguntas y respuestas al uso, para transformarse en un juego emocionante, de acción rápida, divertido y sorprendente, donde las cuestiones que se plantean se utilizan como puente para lograr que el jugador pueda sobrevivir ante la oleada de monstruos.

El uso de un escenario con forma de laberinto, así como vallas o torretas automáticas para defendernos, es considerado una idea bastante prometedora de cara a futuras mejoras o actualizaciones. Este proyecto es tan solo un ejemplo de lo mucho que se puede hacer si seguimos mejorándolo y añadiéndole soporte en un futuro, incrementando las herramientas defensivas, los utensilios, las armas para acabar con los monstruos o incrementando el número de escenarios y niveles.

En general, los usuarios que han podido probar el juego han quedado bastante satisfechos con el resultado. No obstante, debemos decir que el número de personas participantes en la encuesta es tan solo de 5, por lo que las conclusiones obtenidas deben tomarse únicamente a título orientativo y como una primera evaluación.

Respecto a los objetivos iniciales, se han cumplido todos y tan solo quedaría aplicar ciertas mejoras de cara a una posible publicación a través de la tienda de aplicaciones Play Store, para que la aplicación pudiese equipararse con otros juegos de calidad similar y tener el éxito esperado.

En cuanto a las dificultades que han surgido, decir que son las propias de cualquier aplicación que se quiere crear desde cero, sin tener conocimientos anteriormente sobre la materia. Gracias a la ayuda de foros como www.unityspain.com, así como la ayuda proporcionada por la extensa información encontrada en diversos video tutoriales, blogs, foros y libros electrónicos encontrados a lo largo de internet, ha sido posible realizar este proyecto de forma totalmente autónoma y partiendo como único conocimiento previo, el lenguaje de programación Java.

6.2 Trabajos futuros:

Principalmente los trabajos futuros serían poder mejorar algunos detalles de cara a poder publicar el juego en el Play Store con el fin de que la gente lo conociera y pudiera exponer sus valoraciones.

Fundamentalmente para su publicación sería necesaria la utilización de modelos 3D propios que diesen a nuestro juego el toque personal que falta. En el caso de nuestro proyecto, los modelos 3D son importados y totalmente gratuitos por lo que en principio no habría problema en usarlos. En cambio, otros usuarios o desarrolladores, también podrían tener acceso a estos modelos y utilizarlos en sus propias creaciones.

Es por ello que vemos necesario incluir modelos 3D propios y que nos distingan. Para ello tendría que aprenderse las técnicas básicas necesarias para modelado y animación 3D y usar algún software de modelado 3D gratuito como por ejemplo Blender, el cual permite exportar tus creaciones directamente a Unity. Una vez aprendidas estas técnicas, podríamos modelar un personaje principal propio, así como modelos de monstruos distintos a los vistos en el proyecto.

También podríamos añadir elementos como casas u objetos y ampliar el escenario.

Otras posibilidades a nivel de jugabilidad, sería poder incluir nuevas armas u objetos que permitan destruir de forma estratégica a los monstruos. Incluir dinamita que explote cuando sea pisada por los monstruos, añadir torretas más potentes o trampas, así como otros objetos obtenidos a partir de la resolución de preguntas proporcionaría mucha variedad al juego y motivaría aún más al jugador para responder correctamente a las preguntas.

Por otra parte, en cuanto a algunas mejoras que habría que desarrollar de cara a alargar más la duración del juego sería poder ir pasando por diferentes escenarios y dar la posibilidad de desbloquearlos a medida que vas avanzando por los niveles diferentes niveles. De esta forma el juego se estructuraría en diferentes laberintos o escenarios que vamos desbloqueando y por donde el jugador tiene que ingeniárselas para alcanzar la oleada máxima para desbloquear cada escenario. Esto a su vez provocaría que se tuviera que desarrollar en el juego una forma de guardar no solo las puntuaciones, sino los escenarios desbloqueados y los últimos logros alcanzados.

Por último, sería interesante que, puesto que está dirigido en principio a la docencia, pudiera utilizarse para que jugadores pudieran examinarse de sus conocimientos de una determinada materia a la vez que juegan.
Esto podría ir acompañado de la posibilidad de desglosar la puntuación según el número de respuestas acertadas y subirlo a una base de datos o red social. De esta forma se podría consultar la puntuación máxima alcanzada de cada jugador para de esta forma, promocionar el videojuego e incitar a otros amigos o personas cercanas a que se lo descarguen y compitan por conseguir una mayor puntuación.

Capítulo 7: Presupuesto

En este apartado se pretender detallar el presupuesto del proyecto, compuesto de los siguientes apartados a tener en cuenta:

- Recursos materiales empleados: gastos de software y de hardware
- Recursos humanos empleados: gastos del personal

A estos costes se le añadirá un 10% de los mismos en concepto de costes indirectos (electricidad, conexión a internet, etc.) y por último se le sumará el impuesto sobre el valor añadido (IVA) del 18%.

Para realizar todos los cálculos, se parte de la base de que el tiempo total empleado para la realización del proyecto ha sido de 10 meses. En primer lugar vamos a calcular el coste de los recursos materiales y posteriormente calcularemos los recursos humanos.

7.1 Recursos materiales

Para calcular el coste de los recursos materiales, se ha de contar con el presupuesto del coste de hardware, formado por el coste derivado del uso del ordenador de escritorio que se ha utilizado para el desarrollo del proyecto, así como del uso de un dispositivo móvil táctil con sistema operativo Android 4.1.2 para realizar las pruebas pertinentes. Para ello se estima un periodo de amortización de 3 años para el ordenador y 2 años para el dispositivo móvil, haciendo en ambos un uso dedicado para el proyecto de 10 meses.

A estos gastos se añadirán los costes derivados de las licencias de las herramientas software empleadas para su elaboración, cuya vida útil por licencia se estima en 2 años. Algunas como la de Unity 3D se pagan al mes y otras no tienen límite de uso.
En el caso de Unity 3D, se estima el uso de una licencia sin limitaciones, ya que si quisiéramos comercializar la aplicación haciendo uso de Unity Free, no podría ser utilizado por una entidad comercial con ingresos anuales brutos superiores a 100.000 $ ni por una institución educativa académica.

	Ordenador de sobremesa		Dispositivo móvil Android	
	Tiempo	Coste	Tiempo	Coste
Amortización total	36 meses	500 €	24 meses	300 €
Coste en el proyecto	10 meses	138 €	10 meses	125 €

Tabla 33: Costes de hardware

		Licencias	
		Tiempo	Coste
Unity 3D Pro	Amortización total	10 meses	75$/mes -> 55.18€/mes
	Coste en el proyecto	10 meses	552 €
Adobe Photoshop CS4	Amortización total	24 meses	849 €
	Coste en el proyecto	10 meses	354 €
Microsoft Word Professional 2010	Amortización total	Sin limite	539 €
	Coste en el proyecto	10 meses	539 €
Microsoft Visio Professional 2013	Amortización total	Sin limite	739 €
	Coste en el proyecto	10 meses	739 €

Tabla 34: Costes de software

Resursos Materiales	
Coste total de hardware	263 €
Coste total de software	2.184 €
Coste material	2.447 €

Tabla 35: Recursos materiales

7.2 Recursos humanos

Para hacer el cálculo de los recursos humanos se consideran aproximadamente 200 días de trabajo.
Partimos de un tiempo empleado de 10 meses (300 días) en desarrollar el proyecto en jornadas de 4 horas al día de media, de los cuales restamos días festivos (fines de semana 80 días) y vacaciones (navidad y semana santa 17 dias).
Se estima un coste del personal de 30€ /hora brutos

Resursos Humanos	
Días	200 días
Horas	800 horas
Coste humano	24.000 €

Tabla 36: Recursos humanos

En resumen, y considerando el resto de costes comentados al principio de este apartado, el presupuesto final bruto, sin incluir ningún porcentaje de beneficios, sería el siguiente:

Concepto	Coste (Euros)
Recursos Materiales	2.447 €
Recursos Humanos	24.000 €
Costes de los recursos	26.447 €
Costes Indirectos (10%)	2.645 €
Presupuesto sin I.V.A	29.092 €
Costes del I.V.A (18%)	5.236 €
Presupuesto Final	34.328 €

Tabla 37: Presupuesto del Proyecto

Bibliografía

[1] Simone Belli y Cristian López Raventós, Universitat Autònoma de Barcelona, Breve historia de los videojuegos, atheneadigital.net/article/view/570/437 [Última consulta, 13 mayo 2014]

[2]Varios autores. La historia de los videojuegos, *www.elotrolado.net/wiki/Historia_de_los_videojuegos* [Última consulta, 13 mayo 2014]

[3]David Bretos, Juegos para móviles, *http://www.lavanguardia.com/estilos-de-vida/20130524/54373703721/juegos-para-el-movil.html*, [Última consulta, 13 mayo 2014]

[4]Rafael Orduz, Corporación Colombia Digital, Videojuegos: cadena de valor, http://www.colombiadigital.net/opinion/columnistas/los-numeros-de-las-tic/item/1846-videojuegos-cadena-de-valor-ii.htm, [Última consulta, 13 mayo 2014]

[5]Varios autores, AEVI (Asociación Española de videojuegos), www.aevi.org.es/, [Última consulta, 13 mayo 2014]

[6]Varios autores. Videojuegos educativos. Una herramienta para aprender, http://www.todopapas.com/ninos/educacion/videojuegos-educativos-una-herramienta-para-aprender-3859, [Última consulta, 13 mayo 2014]

[7] Gerardo González García, ¿Qué es Software educativo, cómo se clasifica y cuáles son sus características?, http://profesorinteractivo.blogia.com/2007/041701--que-es-software-educativo-.php ,[Última consulta, 13 mayo 2014]

[8]Varios autores, Videojuegos educativos: aprovechando la brecha generacional, http://rt00149b.eresmas.net/pipo.html, [Última consulta, 13 mayo 2014]

[9]Floralicia Anzola, Dragón Box una aplicación innovadora para aprender matemáticas http://www.0800flor.net/tecnologia-e-innovacion/dragon-box/, [Última consulta, 13 mayo 2014]

[10]Didac Arnau, Los 10 mejores videojuegos educativos, *.blog.tiching.com/los-10-mejores-videojuegos-educativos/*, [Última consulta, 13 mayo 2014]

[11] Stencyl, http://www.stencyl.com/, [Última consulta, 13 mayo 2014]

[12] Game Develop, http://www.en.compilgames.net/index.php, [Última consulta, 13 mayo 2014]

[13] Entidad 3D, http://www.entidad-3d.com/, [Última consulta, 13 mayo 2014]

[14] Construct 2, https://www.scirra.com, [Última consulta, 13 mayo 2014]

[15] Kodu Game Lab, http://www.kodugamelab.com/, [Última consulta, 13 mayo 2014]

[16] Unity 3D, http://unity3d.com/es, [Última consulta, 13 mayo 2014]

[17] Lista de juegos con Unity 3D, http://en.wikipedia.org/wiki/List_of_Unity_Engine_games, [Última consulta, 13 mayo 2014]

[18] Tower Defense, http://es.wikipedia.org/wiki/Tower_defense , [Última consulta, 13 mayo 2014]

[19] Edgar Bernal, 2013 Tutorial #1 Unity 3D Introducción, http://www.widget101.com/juegos/unity-3d/introduccion-a-unity-3d/ , [Última consulta, 13 mayo 2014]

ANEXO 1: AYUDA DE USUARIO

Hay que destacar que este apartado se reduce a mostrar la captura de la opción "Tutorial" que se dispone en el juego, ya que es aquí donde el usuario podrá obtener una breve descripción de los elementos que se encontrará en pantalla una vez iniciada la partida:

Figura 83: Tutorial de uso del juego

ANEXO 2: MANUAL EDICIÓN DE PREGUNTAS

En este anexo se pretende indicar al docente los pasos que deberá seguir para editar las cuestiones que se le irán formulando al jugador.

Para ello, el docente no tiene que introducirse en el código del juego ni tener unos conocimientos específicos. Simplemente bastará con hacer las modificaciones sobre el fichero "plantilla.xml", situado en la carpeta "Resources", que será contenedor de las preguntas.

A continuación pasamos a exponer un ejemplo de muestra de dicho fichero con el fin de explicar la estructura que sigue. Vamos a exponer la estructura del fichero con 2 preguntas:

```
<cuestionario>
       <pregunta1>
            <enunciado>¿Cuáles son las partes de una flor?</enunciado>
            <respuesta1>Pétalos, cáliz, sepalos y corona</respuesta1>
            <respuesta2>Pétalos, polen, ovarios y corona</respuesta2>
            <respuesta3>Pétalo, sépalo, antera y tallo</respuesta3>
            <correcta>Pétalos, cáliz, sepalos y corona</correcta>
        </pregunta1>

        <pregunta2>
            <enunciado>¿Que tipo de animal es el  babirusa?</enunciado>
            <respuesta1>Un caracol de Sudamérica</respuesta1>
            <respuesta2>Un insecto de Africa Ecuatorial</respuesta2>
            <respuesta3>Un mamífero de Indonesia</respuesta3>
            <correcta>Un mamífero de Indonesia</correcta>
        </pregunta2>
</cuestionario>
```

Como puede apreciarse, no sigue ningún estándar concreto y su estructura fácil y comprensible para cualquier persona.
Cuestionario es el nodo principal que contendrá todos sus nodos hijo pregunta.
Cada nodo pregunta podemos apreciar que se encuentra compuesto de los siguientes nodos:

- Enunciado: se trata de la pregunta que se le formulará al jugador
- Respuesta 1: primera respuesta posible
- Respuesta 2: segunda respuesta posible
- Respuesta 3: tercera respuesta posible
- Correcta: se trata de la respuesta correcta que deberá coincidir con una de las 3

Los pasos que por tanto ha de seguir el docente será abrir el proyecto con Unity 3D y acceder dentro de **Assets** a la carpeta **Resources**, donde se encontrará el fichero "**plantilla.xml**"

Al hacer doble click sobre el fichero, se abrirá el editor por defecto de Unity: **MonoDevelop** donde podremos realizar las modificaciones oportunas, bien añadiendo más preguntas, modificando las existentes… etc.

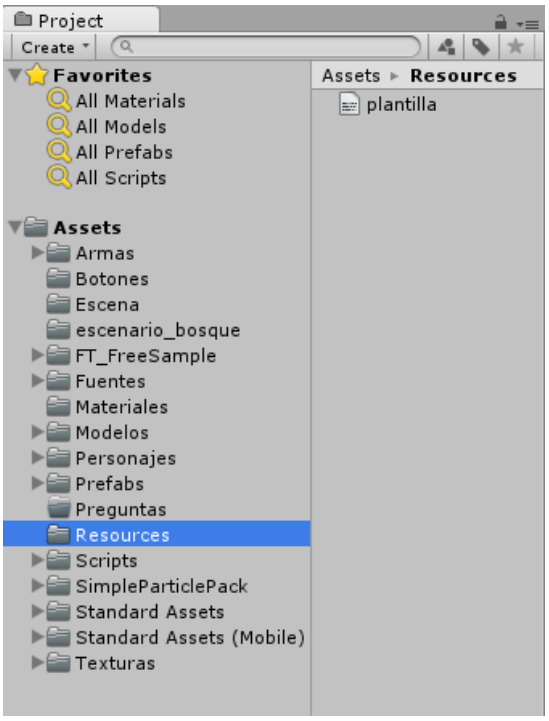


Figura 84: Localización del fichero "plantilla.xml"

Cabe destacar que el sistema no detecta en cada partida qué preguntas han sido o no formuladas, por lo que al ser totalmente aleatorias, podrán aparecer 2 más veces.
Para evitar que se repitan, conviene el docente disponga de un fichero con un número bastante numeroso de preguntas, ya que de este modo habrá menos posibilidad de que salga la misma pregunta varias veces.

Una vez editado y guardado el fichero, el docente deberá compilar el juego en un fichero de tipo APK que será el instalador que posteriormente introduciremos en el dispositivo móvil. Para ello se deberá pinchar en **File > Build Settings…** y se nos mostrará la ventana en la cual podremos seleccionar la escena que compilaremos.
Nuestro juego se encuentra almacenado en una única escena como bien se ha explicado en apartados anteriores, por lo que quedará seleccionada. Una vez hecho esto, procederemos a pulsar el botón "**Build**" e indicaremos el nombre y el directorio donde queremos que quede almacenado nuestro fichero APK.

Recordar que para que pueda compilarse el juego, es necesario tener instalado el SDK de Android.

Para más información, consultar en: http://docs.unity3d.com/Manual/android-sdksetup.html

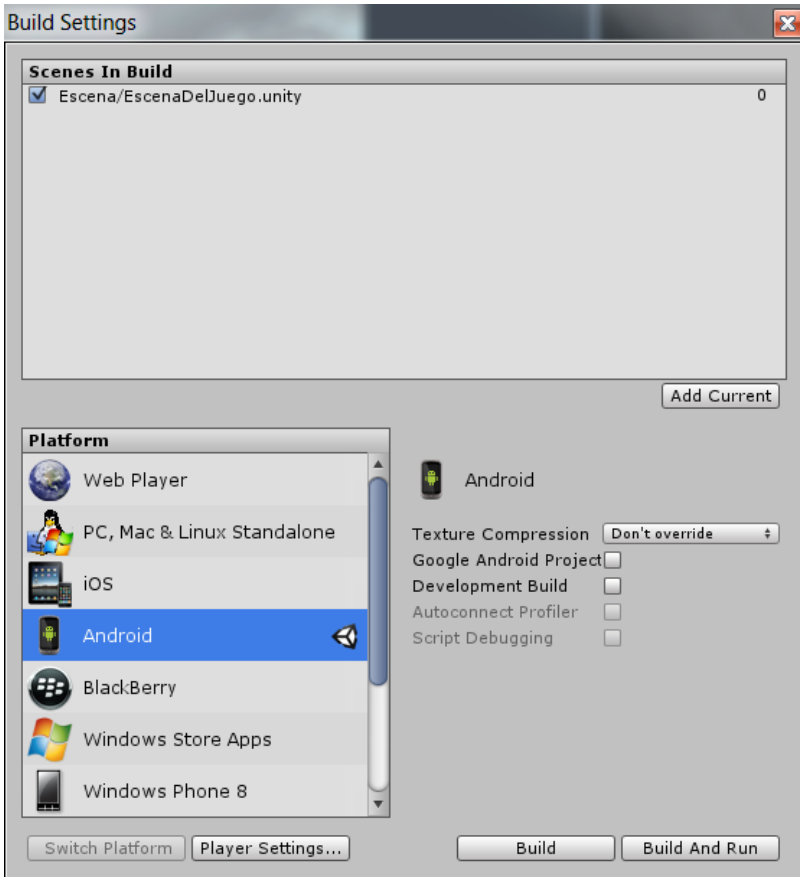


Figura 85: Ventana de Build Settings

ANEXO 3: MANUAL DE INSTALACIÓN

Puesto que el proyecto por el momento no se ha publicado en PlayStore para su descarga, se procederá a explicar cómo hacer la instalación manualmente.

Para ello deberemos de disponer el archivo APK compilado anteriormente.
Bastará con copiar dicho archivo en un directorio fácilmente reconocible de nuestro dispositivo móvil.

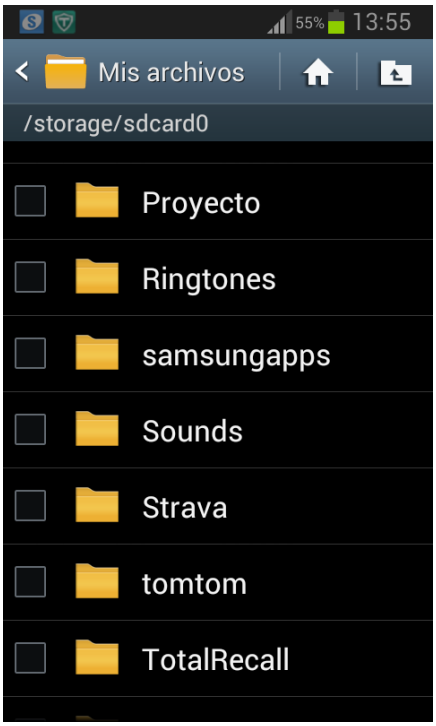


Figura 86: Contenido de la memoria interna del móvil

En el ejemplo hemos creado un directorio llamado "**Proyecto**" donde se encuentra almacenado el fichero APK. Bastará con utilizar el administrador de archivos que trae por defecto nuestro dispositivo móvil y encontrar la carpeta dicho archivo

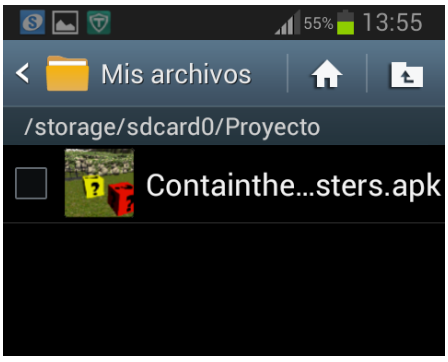


Figura 87: Archivo APK de nuestro proyecto

Una vez hecho pulsada nuestra aplicación, se procederá a instalarse en la memoria interna del móvil. Es importante que haya un espacio libre de al menos 50 mb para aplicaciones.

Figura 88: Procediendo a instalar el proyecto en el dispositivo móvil

Esperamos a que concluya el proceso de instalación.

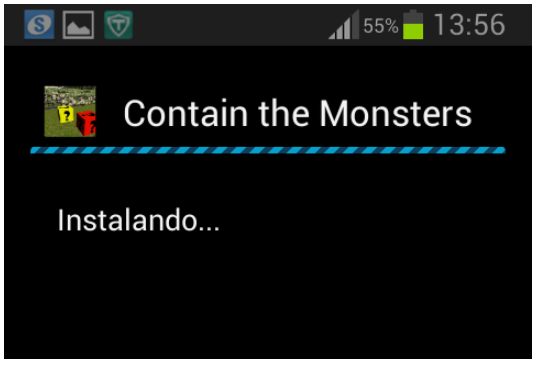


Figura 89: Instalación del proyecto en el dispositivo móvil

Una vez concluido este proceso ya podremos disponer del icono en nuestro menú de aplicaciones que nos permita arrancar el juego.

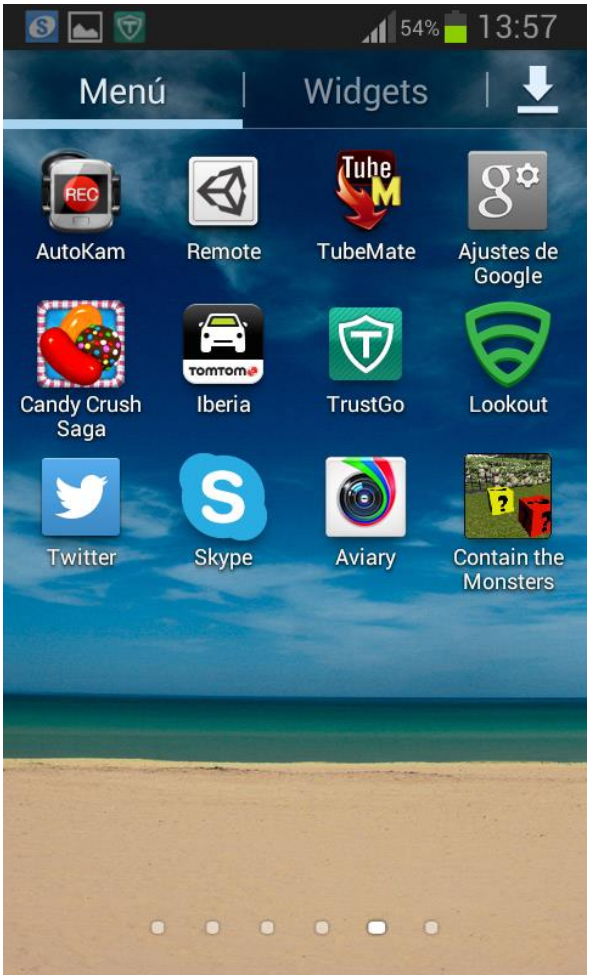


Figura 90: Menú de las aplicaciones instaladas en nuestro móvil